JOURNAL OF COMPUTER AND SYSTEM SCIENCES 25, 285–331 (1982)

Homomorphisms between Models of Parallel Computation*

TAKUMI KASAI

Research Institute for Mathematical Science, Kyoto University, Kyoto 606, Japan

AND

RAYMOND E. MILLER

Georgia Institute of Technology, School of Information and Computer Science, Atlanta, Georgia 30332

Received August 6, 1979; revised April 30, 1982

To better understand the relationships between different models of parallel computation, we introduce a new computation system formulation and develop general notions of homomorphisms and isomorphisms between computation systems. This allows us to study relations between vector addition systems, vector replacement systems, Petri nets, and generalized Petri nets. Results in this paper that may be of particular interest include a long list of properties preserved under homomorphism, and constructions that show that vector replacement systems can be simulated by vector addition systems, and that generalized Petri nets can be emulated by Petri nets.

1. INTRODUCTION

Since the early 1960s many different models of parallel computation have been developed, some being based on finite state machine approaches, others on program schemata models, with many having underlying directed graph representations [1, 2, 8–12, 18–20]. On the surface, these models often appear quite different from one another. Also, the particular types of parallelism problems studied, and the results obtained, seemed often to bear no relation to one another. Thus, the area of modelling parallel computation has seemed rather fragmented.

In attempts to help unify the area, papers have been written comparing the various models and showing relationships between them [2, 10, 13, 15–18]. Thus, results obtained in terms of one model can often be translated into results in another model. It is fairly well known by now, for example, that vector addition systems are in some sense *equivalent* to Petri nets [7, 16, 19], and that some of the other graphical models for parallel computation are somehow *less powerful* than Petri nets or vector addition systems.

* This work was started while Kasai was a Visiting Scientist, and Miller a Research Staff Member, at IBM T. J. Watson Research Center in Yorktown Heights, N.Y. The paper was completed by Miller. The work was partially supported under NSF Research Grant MCS-8103608.

0022-0000/82/060285–47$02.00/0

This paper is also a comparative study of models of parallel computation. Here we introduce a rather general notion of a homomorphism between computation systems, and then use this notion to develop a precise relationship of one computation system being simulated by another computation system. In comparison with previous studies, this approach provides a more precise and deeper understanding of how the properties o the various models are interrelated. Many of the previously discovered relationships follow as corollaries of results obtained here. In particular, the concept of an *isomorphism* between two models turns out to be a particular type of homomorphism which is shown to be both length preserving and bijective.

It is interesting to note that mappings, or homomorphisms, between models of parallel computation have been used elsewhere for somewhat different purposes [5, 12–14], and that there are some similarities between these previous works and the formulations used in this paper. In these other papers, the authors develop homomorphisms that they call *reductions* or *contractions* of systems which are aimed at producing a simplified version of a system which can be used to study the properties of the more complex system and to analyze correctness of the system. We shall discuss how these other formulations are related to our work as we develop our formulations in Sections 2 and 3.

As is often the case in developing theoretical models of computation, especially in the case of parallel computation, the models become very complex, there are many definitions, and a long list of more or less unrelated results. This paper has this character also. It is somewhat difficult to point out the *most important* results, however, since if a reader is interested in the relationships between two particular models it is those results and not the others that are most important to that reader. Nevertheless, as general guidance, the long list of properties that are preserved under particular types of homomorphisms that appears in Section 4 should be highlighted since it is in these results that the exact type of mapping required to preserve a property is specified. Our interest in finding these *preserving* mappings is what leads to the need for the many different types of homomorphisms that we introduce. As one may note from these theorems, spanning homomorphisms play a dominant role.

As for the comparative results between models, the two results that are probably of most interest are that vector replacement systems can be simulated by vector addition systems (Theorem 5.1), and that generalized Petri nets can be emulated by Petri nets (Section 7).

2. Homomorphisms

In this section, we introduce the notion of homomorphisms between computation systems. By means of this notion, we give a precise definition of what it means for one system to simulate another.

We start with some preliminary notation. We denote by Z the set of integers, and by N, the set of natural numbers. Let Σ be any set. We denote by Σ^* the set of finite sequences (strings) of elements of Σ including the null sequence λ. The set of infinite

sequences of elements Σ is denoted by Σ^∞. We denote by Σ^ω the set $\Sigma^\omega = \Sigma^\infty \cup \Sigma^*$. For $x \in \Sigma^\omega$ and $y \in \Sigma^*$, y is said to be a *prefix* of x if there exists a $z \in \Sigma^*$ such that $x = yz$. The length of x is denoted by $|x|$, and if $x \in \Sigma^\infty$ then we let $|x| = \infty$.

We use $x = \langle x_1, ..., x_n \rangle$ to denote an n-tuple. If $x = \langle x_1, ..., x_n \rangle$ and $y = \langle y_1, ..., y_m \rangle$ are tuples, then $x; y$ denotes the $(n+m)$-tuple $\langle x_1, ..., x_n, y_1, ..., y_m \rangle$. The ith component of x is denoted by $x(i)$.

DEFINITION 2.1. A *computation system* S consists of:

(i) a set D,

(ii) an element x of D,

(iii) a finite set Σ of *operations*,

(iv) a function "$\bar{}$" from Σ to the set of partial functions from D to D, that is, for each $a \in \Sigma$, $\bar{a}$ is a partial function from D to D. The function $\bar{}$ is extended to Σ^* by

$$\bar{\lambda} = \text{identity}, \qquad \overline{\alpha\beta}(y) = \bar{\alpha} \cdot \bar{\beta}(y) = \bar{\beta}(\bar{\alpha}(y)), \qquad \alpha \cdot \beta \in \Sigma^*, \ y \in D.$$

We sometimes write $S = (\Sigma, D, x)$ instead of $S = (\Sigma, D, x, \bar{})$.

Here D is intuitively thought of as the set of states of the computation system, where a state includes control information as well as data for synchronization. The element x is then considered to be the initial state of the system. The performance of an operation will create a new state, as defined by the function $\bar{}$, and a sequence of operation performances can be thought of as a computation sequence of the system.

We now provide some further notation on state changes.

Let $S = (\Sigma, D, x)$ be a computation system. For $y \in D$ and $\alpha \in \Sigma^*$, we write $\bar{\alpha}(y) = \perp$ if $\bar{\alpha}(y)$ is undefined, and $\bar{\alpha}(y) \neq \perp$ if $\bar{\alpha}(y)$ is defined. For $\alpha \in \Sigma^*$, $y, z \in D$, the notation

$$y \xrightarrow{\alpha} z$$

means that $\bar{\alpha}(y) = z$, and for $y, z \in D$

$$y \xrightarrow{*} z$$

means that $y \to^\alpha z$ for some $\alpha \in \Sigma^*$.

DEFINITION 2.2. Let $S = (\Sigma, D, x)$ be a computation system. For each $y \in D$, a *computation sequence* from y is a (finite or infinite) sequence of elements of Σ $a_1 a_2 \cdots$ such that

$$\overline{a_1 a_2 \cdots a_i}(y) \neq \perp \qquad \text{for each} \quad i.$$

We denote by $C_s^\omega(y)$ the *set of all computation sequences from* y. We denote by $C_s(y)$ the *set of all finite computation sequences from* y, that is,

$$C_s(y) = \{\alpha \mid \alpha \in \Sigma^*, \bar{\alpha}(y) \neq \perp\}.$$

Note that, for an $\alpha \in \Sigma^{\omega}$, α is in $C_s^{\omega}(y)$ if and only if every prefix of α is in $C_s(y)$. For each $y \in D$, let $R_s(y)$ be the subset of D defined by

$$R_s(y) = \{\bar{a}(y) \mid \alpha \in \Sigma^*, \bar{a}(y) \neq \perp\}.$$

The set $R_s(y)$ is called *the reachability set* from y. When $y = x$, we simply write R_s, C_s, and C_s^{ω} instead of $R_s(x)$, $C_s(x)$, and $C_s^{\omega}(x)$, respectively.

In [12] Kwong defines a *transition system* as a quadruple $(Q, \Sigma, \rightarrow, Q_0)$ to represent a parallel system. This formulation follows that of Lipton [14] and Keller [11]. Although this transition system is similar to our computation system, it differs in several important ways. We let Σ be a finite set of operations, but he does not introduce the concept of operations into his model. Rather, his Σ is a set (possibly infinite) of transitions and $\rightarrow$ is a relation from states and transitions into states, rather than our use of a partial function $\bar{a}(y)$ to indicate what action occurs due to a certain operation a. We feel that the use of operations in the model, and the resulting ability to represent behavior in a deterministic rather than nondeterministic fashion is important for the study of parallel systems. In [5] Gourlay, Rounds, and Statman define *virtual machines* to represent parallel systems, and they include the notion of operations. We shall discuss later how Kwong's use of *reductions* and Gourlay, Rounds, and Statman's notion of *contractions* are related to our homomorphisms.

In later sections, we consider a number of classes of computation systems. To help an understanding of the notion of computation systems, however, we give a more concrete but rather informal example.

EXAMPLE 2.1. Consider the parallel program in Fig. 2.1 discussed in Dijkstra [4], where the initial value for turn is 1.

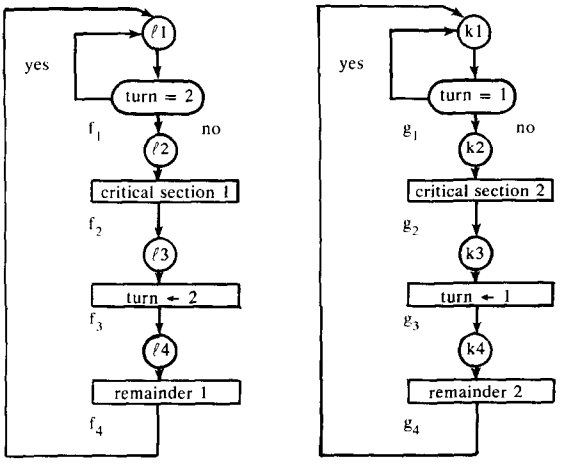


FIG. 2.1. An example of a parallel program.

This program can be formulated as a computation system $S = (\Sigma, D, x)$ defined by

$$\Sigma = \{f_1, f_2, f_3, f_4, g_1, g_2, g_3, g_4\},$$
$$D = \{l1, l2, l3, l4\} \times \{k1, k2, k3, k4\} \times \{1, 2\}, \qquad x = \langle l1, k1, 1\rangle.$$

The partial functions $\bar{f}_i: D \to D$ and $\bar{g}_i: D \to D$ are defined as follows, where $\bar{f}_i(l, k, t)$ and $\bar{g}_i(l, k, t)$, (t = turn), and are undefined if left unspecified:

$$\bar{f}_1(l, k, t) = \langle l2, k, t\rangle, \quad \text{if} \quad l = l1 \quad \text{and} \quad t \neq 2,$$
$$\bar{f}_2(l, k, t) = \langle l3, k, t\rangle, \quad \text{if} \quad l = l2$$
$$\bar{f}_3(l, k, t) = \langle l4, k, 2\rangle, \quad \text{if} \quad l = l3,$$
$$\bar{f}_4(l, k, t) = \langle l1, k, t\rangle, \quad \text{if} \quad l = l4,$$
$$\bar{g}_1(l, k, t) = \langle l, k2, t\rangle, \quad \text{if} \quad k = k1 \quad \text{and} \quad t \neq 1,$$
$$\bar{g}_2(l, k, t) = \langle l, k3, t\rangle, \quad \text{if} \quad k = k2,$$
$$\bar{g}_3(l, k, t) = \langle l, k4, 1\rangle, \quad \text{if} \quad k = k3,$$
$$\bar{g}_4(l, k, t) = \langle l, k1, t\rangle, \quad \text{if} \quad k = k4.$$

Then

$$\langle l1, k1, 1\rangle \xrightarrow{f_1} \langle l2, k1, 1\rangle \xrightarrow{f_2} \langle l3, k1, 1\rangle \xrightarrow{f_3} \langle l4, k1, 2\rangle$$
$$\xrightarrow{g_1} \langle l4, k2, 2\rangle \xrightarrow{g_2} \langle l4, k3, 2\rangle \xrightarrow{f_4} \langle l1, k3, 2\rangle$$

is a computation of S. Hence $f_1 f_2 f_3 g_1 g_2 f_4$ is in C_S. The reachability set of S is

$$R_S = \{\langle l, k, t\rangle \mid t = 1, k = k1 \text{ or } k = k4\} \cup \{\langle l, k, t\rangle \mid t = 2, l = l1 \text{ or } l = l4\}.$$

Note that $\bar{f}_1(l1, k, 2)$ and $\bar{g}_1(l, k1, 1)$ we left undefined even though they could be considered to be defined as equal to $\langle l1, k, 2\rangle$ and $\langle l, k1, 1\rangle$, respectively. It turns out that this leads to no difficulties in the analysis for this example.

DEFINITION 2.3. Let $S_1 = (\Sigma_1, D_1, x_1)$ and $S_2 = (\Sigma_2, D_2, x_2)$ be computation systems. A *homomorphism* $h: S_1 \to S_2$ consists of a homomorphism $\tau: \Sigma_1^* \to \Sigma_2^*$, and an injection $\rho: D_1 \to D_2$ which satisfies the following conditions:

$$\rho(x_1) = x_2 \tag{2.1}$$

$$\text{for each } y, z \in R_{S_1} \text{ and } \alpha \in \Sigma_1^*, \text{ if } y \xrightarrow{\alpha} z, \text{ then } \rho(y) \xrightarrow{\tau(\alpha)} \rho(z). \tag{2.2}$$

A homomorphism is said to be *injective* if $\tau: \Sigma_1^* \to \Sigma_2^*$ is injective. A homomorphism is said to be *surjective* if the following conditions hold:

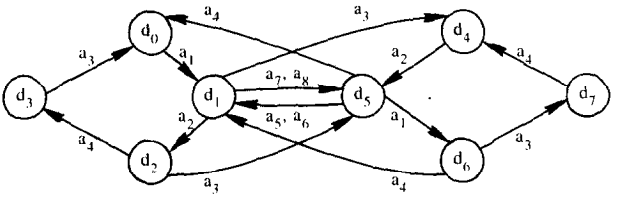


FIG. 2.2. The state transition diagram of S'.

for any $y, z \in R_{S_1}$ and $\beta \in \Sigma_2^*$, if $\rho(y) \xrightarrow{\beta} \rho(z)$, then there exists an $\alpha \in \Sigma_1^*$ such that $\beta = \tau(\alpha)$ and $y \xrightarrow{\alpha} z$. (2.3)

for any $y \in R_{S_1}$ and $z' \in D_2$, if $\rho(y) \xrightarrow{*} z'$, then $z' \xrightarrow{*} \rho(z)$ for some $z \in R_{S_1}$. (2.4)

The injection ρ relates elements of D_1 to elements of D_2, where condition (2.1) states that the starting state of S_1 maps into the starting state of S_2. Also, by condition (2.2), if α is any computation sequence in S_1 then there is a related computation sequence $\tau(\alpha)$ in S_2, with appropriate state mappings using ρ. Condition (2.3) means that any computation from $\rho(y)$ to $\rho(z)$ in S_2 is the image of a computation from y to z in S_1. Condition (2.4) means that any computation from $\rho(y)$ in S_2 is an initial segment of the image of some computation from y in S_1.

A homomorphism is called *bijective* if it is surjective as well as injective.

EXAMPLE 2.2. Consider the computation system S in Example 2.1. Since $\#D$ is finite[1], S can be regarded as a finite state machine (without final state). The state transition diagram, however consists of 32 states and 56 edges. Let $S' = (\Sigma', D', d_0)$ be the computation system defined by

$$\Sigma' = \{a_1, a_2, ..., a_8\}, \qquad D' = \{d_0, d_1, ..., d_7\},$$

and for each i, the partial function $\bar{a}_i$ is defined as in Fig. 2.2, where $d_j \to^{a_i} d$, means that $\bar{a}_i(d_j) = d_l$. Let ρ: $D' \to D$ be the function defined by

$$\rho(d_0) = \langle l1, k1, 1\rangle, \qquad \rho(d_4) = \langle l1, k1, 2\rangle,$$
$$\rho(d_1) = \langle l4, k1, 2\rangle, \qquad \rho(d_5) = \langle l1, k4, 1\rangle,$$
$$\rho(d_2) = \langle l4, k4, 1\rangle, \qquad \rho(d_6) = \langle l4, k4, 2\rangle,$$
$$\rho(d_3) = \langle l4, k1, 1\rangle, \qquad \rho(d_7) = \langle l1, k4, 2\rangle.$$

[1] For a set D, $\#D$ denotes the cardinality of D. This notation is used since $|s|$ is already used to indicate sequence length.

Let τ: $(\Sigma')^* \to \Sigma^*$ be the homomorphism defined by

$$\tau(a_1) = f_1 f_2 f_3, \quad \tau(a_2) = g_1 g_2 g_3, \quad \tau(a_3) = f_4, \quad \tau(a_4) = g_4,$$
$$\tau(a_5) = f_1 g_4 f_2 f_3, \quad \tau(a_6) = f_1 f_2 g_4 f_3, \quad \tau(a_7) = g_1 f_4 g_2 g_3, \quad \tau(a_8) = g_1 g_2 f_4 g_3.$$

It should be clear that $h = (\tau, \rho)$ is a bijective homomorphism from S' to S. Many of the properties of S can thus be studied through S', a much simpler computation system.

As is clear from this example, our homomorphism formulation provides a mapping from the *simpler* system S' to the *more complex* system S. The *reductions* of Kwong [12] and the *contractions* of Gourlay *et al.* [5], just by the names themselves, are clearly mappings from the *more complex* to the *simpler* system. This is natural for the types of purposes for which [5 and 12] are using the mappings, but for our comparative analysis of models, we find that the direction of the mapping we use provides more flexibility and generality than in [5 or 12]. For example, the basic definition of reduction [12, Definition 4.1] has conditions which seem to restrict reductions to what we call surjective homomorphisms. Also, the reduction notion does not relate sequences of transitions between S and S', that is, the τ mapping is not used in reductions. Thus, under reductions one loses all information about what sequence of transitions actually occurred in the system. If we now consider a further definition of reduction [12, Definition 5.1], this requires a pair of mappings T and T' going in opposite directions between the systems. In this case T' is very much like our τ, but here it appears that the induction can only handle reducing strictly sequential nonbranching portions of the systems to single actions in the simplified system. In this case, a single transition of the simplified systems would correspond to only one sequence of actions in the original system so there would be no ambiguity created in the reduction as to which transition sequence occurred.

In [5], Gourlay *et al.* show how their notion of contractions relates to reductions. As in the case of reductions, the contractions differ from our homomorphisms in the direction of the mapping and in the sense that contractions contain no mapping between sequences of actions between the two systems. Possibly more important than these technical differences between our homomorphisms and the reduction and contraction notions, however, are the differences in the aims of the studies. We are primarily interested in relationships between different models of parallel computation, and in what properties carry over from one model to the next, whereas the emphasis in [5, 12] is to aid in the analysis of systems. Of course, for both of these aims one is interested in properties that are preserved under the mappings, and this forms the common bond between the studies.

DEFINITION 2.4. Let $S = (\Sigma, D, x)$ be a computation system, and let α and β be elements of Σ^*. We write $\alpha \simeq \beta$ if α is a permutation of β. A homomorphism h: $S_1 \to S_2$ is said to be *spanning* if the following conditions hold:

for any $y, z \in R_{S_1}$ and $\beta \in \Sigma_2^*$, if $\rho(y) \xrightarrow{\beta} \rho(z)$, then there exists an

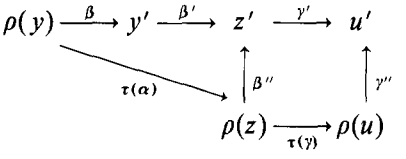
$\alpha \in \Sigma_1^*$ such that $y \xrightarrow{\alpha} z$ and $\beta \simeq \tau(\alpha)$. (2.5)

there exists a $k \in N$ such that for $y \in R_{S_1}$, $y' \in D_2, \beta \in \Sigma_2^*$, if $\rho(y) \xrightarrow{\beta} y'$, then there exist $\alpha \in \Sigma_1^*, z \in D_1$, $z' \in D_2, \beta', \beta'' \in \Sigma_2^*$ such that

$$\rho(y) \xrightarrow{\tau(\alpha)} \rho(z) \xrightarrow{\beta''} z', \qquad y' \xrightarrow{\beta'} z', \qquad \tau(\alpha)\beta'' \simeq \beta\beta', \qquad |\beta''| \leqslant k. \quad (2.6)$$

Furthermore, if $z \to^{\gamma} u$ in S_1, then there exist $u' \in D_2$, $\gamma', \gamma'' \in \Sigma_2^*$ such that

$$z' \xrightarrow{\gamma'} u', \qquad \rho(u) \xrightarrow{\gamma''} u', \qquad \tau(\gamma)\gamma'' \simeq \beta''\gamma'.$$



Condition (2.6) means that any computation β from $\rho(y)$ of S_2 is a prefix of a permutation of a computation of the form $\tau(\alpha)\beta''$, $\alpha \in \Sigma_1^*$, $|\beta''| \leqslant k$. Furthermore, if $\alpha\gamma$ is a computation from y of S_1, then there must be a computation of the form $\beta\beta'\gamma'$ such that $\tau(\alpha)\,\tau(\gamma)$ is a prefix of a permutation of $\beta\beta'\gamma'$. Intuitively, β is an initial segment of a simulation of $\tau(a)$, and if $\alpha\gamma$ is a computation of S_1, then β can be followed by a computation to simulate $\tau(\gamma)$.

LEMMA 2.1. *Every surjective homomorphism is a spanning homomorphism.*

Proof. Condition (2.5) follows from (2.3). Condition (2.6) follows from (2.3) and (2.4). In this case, $k = 0$, $\beta'' = \lambda$, $\rho(z) = z'$. ∎

EXAMPLE 2.3. Consider the computation system $S = (\Sigma, D, x)$ in Example 2.1. Let $S'' = (\{a, b\}, \{d_0, d_1\}, d_0)$ be the computation system defined in Fig. 2.3.

Let ρ: $\{d_0, d_1\} \to D$ and τ: $\{a,b\}^* \to \Sigma^*$ be defined by

$$\rho(d_0) = \langle l1, k1, 1\rangle, \qquad \rho(d_1) = \langle l1, k1, 2\rangle, \qquad \tau(a) = f_1 f_2 f_3 f_4, \qquad \tau(b) = g_1 g_2 g_3 g_4.$$

Clearly $h = (\rho, \tau)$ is a homomorphism from S'' to S.

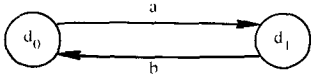


FIG. 2.3. The state transition diagram of S''.

Let $\beta = f_1 f_2 f_3 g_1 g_2 f_4 g_3 g_4$. Then

$$\rho(d_0) = \langle l1, k1, 1\rangle \xrightarrow{\beta} \langle l1, k1, 1\rangle = \rho(d_0),$$

is a computation in S. Since there is no $\alpha \in \{a, b\}^*$ such that $\tau(\alpha) = \beta$, h is not surjective. Now we show that h is a spanning homomorphism. Suppose that $\rho(d_i) \to^{\beta} \rho(d_j)$, $i, j \in \{0, 1\}$. We prove (2.5) by induction on the length of β. Clearly (2.5) holds for $\beta = \lambda$. Suppose that $|\beta| > 0$ and (2.5) holds for any computation of length less than $|\beta|$. Suppose that $i = 0$. Then β must be of the form

$$\beta = f_1 f_2 f_3 \beta_1 f_4 \beta_2, \qquad \beta_1 \in \{g_1, g_2, g_3, g_4\}^*, \quad \beta_2 \in \Sigma^*.$$

Then there exists a computation of the form

$$\rho(d_0) = \langle l1, k1, 1\rangle \xrightarrow{f_1 f_2 f_3 f_4} \langle l1, k1, 2\rangle = \rho(d_1) \xrightarrow{\beta_1 \beta_2} \rho(d_j),$$

since for $y, z \in D$, $y \to^{f_4 \beta_1} z$ implies $y \to^{\beta_1 f_4} z$. By analogous reasoning, (2.5) holds for $i = 1$. Also, it should be clear that (2.4) holds. Hence, (2.6) follows from (2.4) and (2.5).

Thus $h = (\rho, \tau)$ is a spanning homomorphism from S'' to S. As we shall see later, this allows us to study the *behavior* of S through analyzing the simpler system S''. That $h = (\rho, \tau)$ is assumed in much of what follows.

The following lemma is an immediate consequence of Definition 2.3.

LEMMA 2.2. *If* h: $S_1 \to S_2$ *is a homomorphism, then* $\rho(R_{S_1}) \subseteq R_{S_2}$ *and* $\tau(C_{S_1}) \subseteq C_{S_2}$.[2]

A homomorphism τ: $\Sigma_1^* \to \Sigma_2^*$ can be extended to the function τ: $\Sigma_1^\omega \to \Sigma_2^\omega$ in the natural way, that is, for each $a_1 a_2 \cdots$ in Σ_1^ω,

$$\tau(a_1 a_2 \cdots) = \tau(a_1)\, \tau(a_2) \cdots.$$

COROLLARY 2.1. *If* h: $S_1 \to S_2$ *is a homomorphism, then* $\tau(C_{S_1}^\omega) \subseteq C_{S_2}^\omega$.

Proof. Let $\alpha \in C_{S_2}^\omega$. If α is finite, then $\tau(\alpha) \in \tau(C_{S_1}) \subseteq C_{S_2}$. Suppose that α is infinite. Let β' be any prefix of $\tau(\alpha)$. Then there exists a prefix α' of α such that β' is a prefix of $\tau(\alpha')$. Since $\tau(\alpha') \in C_{S_2}$, $\beta' \in C_{S_2}$. That is, every prefix of $\tau(\alpha)$ is in C_{S_2}. Hence $\tau(\alpha)$ is in $C_{S_2}^\omega$. ∎

We now give further results relating R_{S_1} with R_{S_2} and C_{S_1} and C_{S_2} under particular types of homomorphisms.

A homomorphism h: $S_1 \to S_2$ is said to be *length preserving* if $|\tau(a)| = 1$ for all $a \in \Sigma_1$.

[2] Let f: $X \to Y$ be a function. For each $A \subseteq X$, $f(A)$ denotes the set $f(A) = \{f(x) \mid x \in A\}$.

LEMMA 2.3. *Suppose h is a homomorphism from S_1 to S_2 such that for all $a \in \Sigma_1$, $\tau(a) \neq \lambda$. Then $\tau(\alpha) = \beta_1\beta_2$ implies there exist α_1 and α_2 such that $\tau(\alpha_1) = \beta_1$, $\tau(\alpha_2) = \beta_2$, and $\alpha = \alpha_1\alpha_2$ if and only if h is length preserving.*

Proof. Clearly if h is length preserving, then τ satisfies this condition. Conversely, suppose that τ satisfies this condition. Let a be any element of Σ_1 and let $\tau(a) = b\beta$, $b \in \Sigma_2$, $\beta \in \Sigma_2^*$. Then there exist α_1, α_2 such that $\tau(\alpha_1) = b$, $\tau(\alpha_2) = \beta$, and $a = \alpha_1\alpha_2$. Since $\tau(\alpha_1) \neq \lambda$, $\alpha_1 = a$ and $\alpha_2 = \lambda$. Thus $\tau(\alpha_2) = \lambda = \beta$. Hence $\tau(a) = b \in \Sigma_2$. Therefore h is length preserving. ∎

LEMMA 2.4. *If h: $S_1 \to S_2$ is a length preserving surjective homomorphism, then $\rho(R_{S_1}) = R_{S_2}$.*

Proof. Let $S_i = (\Sigma_i, D_i, x_i)$, $i = 1, 2$. Let y' be an arbitrary element of R_{S_2}. Since $\rho(x_1) = x_2 \to^* y'$, it follows from (2.4) that there exists $z \in R_{S_1}$ such that $y' \to^* \rho(z)$. Let β_1 and β_2 be elements of Σ_2^* such that

$$\rho(x_1) = x_2 \xrightarrow{\beta_1} y' \xrightarrow{\beta_2} \rho(z).$$

By (2.3), there exists $\alpha \in \Sigma_1^*$ such that $x_1 \to^\alpha z$ and $\tau(\alpha) = \beta_1\beta_2$. Since τ is length preserving, there exist α_1 and α_2 in Σ_1^* such that $\tau(\alpha_1) = \beta_1$ and $\tau(\alpha_2) = \beta_2$. Let y be the element of D_1 such that $x_i \to^{\alpha_1} y$. Then $\rho(y) = y'$. Hence $y' \in \rho(R_{S_1})$, so $R_{S_2} \subseteq \rho(R_{S_1})$. By Lemma 2.2, $\rho(R_{S_1}) \subseteq R_{S_2}$, so $\rho(R_{S_1}) = R_{S_2}$. ∎

LEMMA 2.5. *If h: $S_1 \to S_2$ is a length preserving surjective homomorphism, then $\tau(C_{S_1}) = C_{S_2}$.*

Proof. Let β be an arbitrary element of C_{S_2}. Then $x_2 \to^\beta y'$ for some $y' \in R_{S_2}$. By Lemma 2.4, $y' = \rho(y)$ for some $y \in R_{S_1}$. Thus, by (2.3), there exists an $\alpha \in \Sigma_1^*$ such that $x_1 \to^\alpha y$ and $\tau(\alpha) = \beta$. Therefore $\beta \in \tau(C_{S_1})$, so $C_{S_2} \subseteq \tau(C_{S_1})$. Applying Lemma 2.2, we get $\tau(C_{S_1}) = C_{S_2}$. ∎

COROLLARY 2.2. *If h: $S_1 \to S_2$ is a length preserving surjective homomorphism, then $\tau(C_{S_1}^\omega) = C_{S_2}^\omega$.*

Proof. Let $\beta \in C_{S_2}^\omega$. If β is finite, then $\beta \in C_{S_2} = \tau(C_{S_1}) \subseteq \tau(C_{S_1}^\omega)$. Suppose that $\beta = b_1 b_2 \cdots$ is infinite. Then, for each i, there exists $a_i \in \Sigma_1$ such that $\tau(a_i) = b_i$ and $a_1 a_2 \cdots a_i \in C_{S_1}$. Thus $\alpha = a_1 a_2 \cdots$ is in $C_{S_1}^\omega$ and $\tau(\alpha) = \beta$. ∎

THEOREM 2.1. *Let h: $S_1 \to S_2$ be a length preserving homomorphism, then h is surjective if and only if $\rho(R_{S_1}) = R_{S_2}$ and $\tau(C_{S_1}) = C_{S_2}$.*

Proof. The *only if* portion follows from Lemmas 2.4 and 2.5. We now show the *if* portion. Suppose that $\rho(R_{S_1}) = R_{S_2}$ and $\tau(C_{S_1}) = C_{S_2}$.

Suppose that $\rho(y) \to^{\beta} \rho(z)$ for $y, z \in R_{S_1}$ and $\beta \in \Sigma_2^*$. Since $\rho(y)$ is in R_{S_2}, there exists $\gamma \in \Sigma_2^*$ such that

$$x_2 = \rho(x_1) \xrightarrow{\gamma} \rho(y) \xrightarrow{\beta} \rho(z).$$

Then $\gamma\beta \in C_{S_2}$. Hence there exists $\alpha \in C_{S_1}$ such that $\tau(\alpha) = \gamma\beta$. Since τ is length preserving, there exist α_1 and α_2 such that $\tau(\alpha_1) = \gamma$, $\tau(\alpha_2) = \beta$. Since $\alpha_1\alpha_2$ is in C_{S_1}, there exist y' and z' in R_{S_1} such that

$$x_1 \xrightarrow{\alpha_1} y' \xrightarrow{\alpha_2} z'.$$

By (2.2), $\rho(y) = \rho(y')$ and $\rho(z) = \rho(z')$. Since ρ is injective, $y = y'$ and $z = z'$. Therefore

$$y \xrightarrow{\alpha_2} z \qquad \text{and} \qquad \tau(\alpha_2) = \beta.$$

Thus (2.3) holds. Since $\rho(R_{S_1}) = R_{S_2}$, (2.4) holds. Therefore h is surjective. ∎

3. Isomorphisms

As we will see in later sections, many of the properties of interest for computation systems are expressed in terms of the R_S and C_S sets. Thus, the results we have just obtained in relating these sets through homomorphisms will provide the necessary tools for showing the relationships between the properties of the different models. Of particular interest, of course, is the case when two different models can be considered *isomorphic*. This notion is developed next.

Lemma 3.1. *Let h be a bijective homomorphism from $S_1 = (\Sigma_1, D_1, x_1)$ to $S_2 = (\Sigma_2, D_2, x_2)$. Then, for each $y, z \in R_{S_1}$ and $\alpha \in \Sigma_1^*$,*

$$y \xrightarrow{\alpha} z \quad \textit{iff} \quad \rho(y) \xrightarrow{\tau(\alpha)} \rho(z). \tag{3.1}$$

Proof. Immediately follows from Definition 2.3. ∎

Lemma 3.2. *Let $\tau: \Sigma_1^* \to \Sigma_2^*$ be a homomorphism. Then $\tau: \Sigma_1^* \to \Sigma_2^*$ is bijective if and only if τ is length preserving and the restriction $\tau \mid \Sigma_1 : \Sigma_1 \to \Sigma_2$ is bijective.*

Proof. Evident. ∎

Definition 3.1. A homomorphism $h: S_1 \to S_2$ is called an *isomorphism* if there is a homomorphism $h': S_2 \to S_1$ such that $hh': S_2 \to S_2$ and $h'h: S_1 \to S_1$ are identities; that is, $\tau\tau': \Sigma_2^* \to \Sigma_2^*$, $\tau'\tau: \Sigma_1^* \to \Sigma_1^*$, $\rho\rho': D_2 \to D_2$, $\rho'\rho; D_1 \to D_1$ are identities.

LEMMA 3.3. *If h: $S_1 \to S_2$ is an isomorphism, then h is a length preserving bijective homomorphism.*

Proof. Let h be an isomorphism. Then there exists a homomorphism h': $S_2 \to S_1$ such that $\tau\tau'$, $\tau'\tau$, $\rho\rho'$, $\rho'\rho$ are identities. Since $\tau\tau'$ and $\tau'\tau$ are identities, τ is bijective. By Lemma 3.2, τ is length preserving. Hence h is length preserving. By Lemma 2.2, $\rho'(R_{S_2}) \geqslant R_{S_1}$ and $\tau'(C_{S_1}) \subseteq C_{S_2}$. Then

$$R_{S_2} = \rho(\rho'(R_{S_2})) \subseteq \rho(R_{S_1}) \subseteq R_{S_2}, \qquad C_{S_2} = \tau(\tau'(C_{S_2})) \subseteq \tau(C_{S_1}) \subseteq C_{S_2}.$$

Thus, $\rho(R_{S_1}) = R_{S_2}$ and $\tau(C_{S_1}) = C_{S_2}$. Therefore, by Theorem 2.1, h is surjective. Since τ is injective, h is injective. ∎

COROLLARY 3.1. *Let $S_i = (\Sigma_i, D_i, x_i)$, $i = 1, 2$ be computation systems. Let τ: $\Sigma_1^* \to \Sigma_2^*$ be a homomorphism and ρ: $D_1 \to D_2$ be a function. Then $h = (\tau, \rho)$ is an isomorphism from S_1 to S_2 if and only if τ and ρ are bijective and satisfy conditions* (2.1) *and* (3.1).

Proof. (only if) Suppose that h is an isomorphism. Clearly, τ and ρ are bijections. By Lemma 3.3, h is a bijective homomorphism, so (3.1) holds.

(if) Let τ': $\Sigma_2^* \to \Sigma_1^*$ and ρ': $D_2 \to D_1$ be defined by $\tau' = \tau^{-1}$ and $\rho' = \rho^{-1}$, respectively. By (3.1), for each y, $z \in R_{S_2}$ and $\alpha \in \Sigma_2^*$, if $y \to^{\alpha} z$, then $\rho'(y) \to^{\tau'(\alpha)} \rho'(z)$. Thus, $h' = (\tau', \rho')$ is a homomorphism from S_2 to S_1. Clearly, $\tau\tau'$, $\tau'\tau$, $\rho\rho'$, $\rho'\rho$ are identities. Therefore, h is an isomorphism. ∎

DEFINITION 3.2. Let $S = (\Sigma, D, x)$ be a computation system. An operation $a \in \Sigma$ is called a *partial identity* if $\bar{a}(y) = y$ or $\bar{a}(y) = \perp$ for all $y \in R_S$.

Hereafter we assume that a computation system contains no partial identity operation.

LEMMA 3.4. *If h: $S_1 \to S_2$ is a homomorphism, then for all $a \in \Sigma_1$, $\tau(a) \neq \lambda$.*

Proof. Suppose that $\tau(a) = \lambda$ for some $a \in \Sigma$. Suppose that $\bar{a}(y) = z$ for some $y \in R_{S_1}$. Then $\rho(y) \to^{\lambda} \rho(z)$. Thus $\rho(y) = \rho(z)$. Since ρ is an injection, $y = z$. Therefore a must be a partial identity, which is excluded by assumption. ∎

DEFINITION 3.3. A computation system $S = (\Sigma, D, x)$ is said to be *reduced* if $D = R_S$. For each computation system $S = (\Sigma, D, x)$, the computation system $\hat{S}$ defined by $\hat{S} = (\Sigma, R_S, x)$ is called *the reduced form of S*, where for each $a \in \Sigma$, the partial function $\bar{a}$ in $\hat{S}$ is the restriction $\bar{a} \mid R_S$ of $\bar{a}$ in S.

THEOREM 3.1. *Let S_1 and S_2 be reduced computation systems. Then h: $S_1 \to S_2$, is an isomorphism if and only if h is length preserving and bijective.*

Proof. By Lemma 3.3, it suffices to show the *if* portion. Since h is surjective, it

follows from Lemma 2.4, that $\rho(D_1)=\rho(R_{S_1})=R_{S_2}=D_2$. Thus, ρ: $D_1\to D_2$ is bijective. Let b be an arbitrary element of Σ_2. Since b is not a partial identity, $\bar{b}(y')\neq\perp$ for some $y'\in D_2$. Then there exist $y, z\in D_1$ such that

$$y'=\rho(y)\xrightarrow{b}\rho(z).$$

Since h is surjective, $b=\tau(\alpha)$ for some $\alpha\in\Sigma_1^*$. Thus τ: $\Sigma_1^*\to\Sigma_2^*$ is surjective. Since h is injective τ: $\Sigma_1^*\to\Sigma_2^*$ is injective. Therefore τ: $\Sigma_1^*\to\Sigma_2{}^*$ is bijective.

Since h is a bijective homomorphism, conditions (2.1) and (3.1) hold. Therefore, by Corollary 3.1, h is an isomorphism. ∎

DEFINITION 3.4. Let Σ be a finite set. For each $w\in\Sigma^*$, let $\iota(w)$ be the subset of Σ defined by

$$\iota(w)=\{a\mid \alpha a\beta=w, \alpha,\beta\in\Sigma^*\}.$$

A surjective homomorphism h: $S_1\to S_2$ is said to be *principal* if for each a and b in Σ_1, either $\tau(a)=\tau(b)$ or $\iota(\tau(a))\cap\iota(\tau(b))=\varnothing$, and $\bar{a}\neq\bar{b}$ implies $\iota(\tau(a)\cap\iota(\tau(b))=\varnothing$.

Principal homomorphisms play a central role in the comparitive study in this paper. In particular, one should note that every isomorphism is a principal homomorphism.

LEMMA 3.5. *Let h: $S_1\to S_2$ be a principal homomorphism. Let $\tau(a)=b\gamma$, $a\in\Sigma_1$, $b\in\Sigma_2$, $\gamma\in\Sigma_2^*$. Then for $y\in R_{S_1}$,*

$$\bar{a}(y)\neq\perp \quad \textit{iff} \quad \overline{\tau(a)}(\rho(y))\neq\perp$$
$$\textit{iff} \quad \bar{b}(\rho(y))\neq\perp.$$

Proof. Clearly, $\bar{a}(y)\neq\perp$ implies $\overline{\tau(a)}(\rho(y))\neq\perp$, and $\overline{\tau(a)}(\rho(y))\neq\perp$ implies $\bar{b}(\rho(y))\neq\perp$. Suppose that $\bar{b}(\rho(y))\neq\perp$. Then there exists a $z'\in D_2$ such that

$$\rho(y)\xrightarrow{b}z'.$$

By (2.4), there exist $z\in D_1$ and $\gamma'\in\Sigma_2^*$ such that

$$\rho(y)\xrightarrow{b}z'\xrightarrow{\gamma'}\rho(z).$$

By (2.3), there *exist* $a'\in\Sigma_1$ and $\alpha'\in\Sigma_1^*$ such that

$$y\xrightarrow{a'\alpha'}z \qquad \text{and} \qquad \tau(a'\alpha')=b\gamma'.$$

Since $b\in\iota(\tau(a))\cap\iota(\tau(a'))$, $\tau(a)=\tau(a')$ so that $\bar{a}=\bar{a}'$. Hence $\bar{a}(y)$ is defined. ∎

COROLLARY 3.2. *Let $h: S_1 \to S_2$ be a principal homomorphism. Then for $\alpha \in \Sigma_1^*$ and $y \in R_{S_1}$,*

$$\bar{\alpha}(y) \neq \perp \quad \text{iff} \quad \overline{\tau(\alpha)}(\rho(y)) \neq \perp.$$

Hence, h satisfies Condition (3.1).

DEFINITION 3.5. Let $S_i = (\Sigma_i, D_i, x_i)$, $i = 1, 2$, be computation systems with D_i being partially ordered sets with the order $\leqslant$. A homomorphism $h: S_1 \to S_2$ is said to be *order preserving* if the following conditions are satisfied:

$$\text{for each } y \text{ and } z \text{ in } D_1,\ y \leqslant z \text{ iff } \rho(y) \leqslant \rho(z), \tag{3.2}$$

$$\text{for each } y \in \rho(D_1) \text{ and } z \in R_{S_2} - \rho(D_1),\ y \not\leqslant z \text{ and } z \not\leqslant y. \tag{3.3}$$

DEFINITION 3.6. Let S_1 and S_2 be computation systems. Suppose that there is a homomorphism $h: S_1 \to S_2$. We say that S_1 is *simulated by* S_2 with respect to h. Alternatively, we say that S_2 is *emulated by* S_1 if there is a homomorphism h: $S_1 \to S_2$.

Let C_1 and C_2 be classes of computation systems and let H be a class of homomorphisms. We say that C_1 is *simulated by* C_2 with respect to H if and only if for any $S_1 \in C_1$, we can effectively construct $S_2 \in C_2$ and a homomorphism $h \in H$ such that $h: S_1 \to S_2$. We say that C_1 is *emulated by* C_2 with respect to H if for any $S_1 \in C_1$, we can effectively construct $S_2 \in C_2$ and $h \in H$ such that $h: S_2 \to S_1$.

Note that for S_1 simulated by S_2, the homomorphism specifies that a computation sequence α of S_1 is simulated by $\tau(\alpha)$ of S_2. It may be that $\tau(\alpha)$ is longer than α. In this case, S_2 would be thought of as requiring extra steps to perform the simulation. In a similar manner, in emulation $h: S_2 \to S_1$, S_2 may emulate a sequence in S_1 in fewer steps than S_1 requires.

EXAMPLE 3.1. Consider the parallel program described in Fig. 3.1, where the initial values for s and buffer are $s = 1$ and buffer $= 0$. We assume that each operation (i.e., each box in the figure) is indivisible.

This program can be formulated as the computation system $S_1 = (\Sigma_1, D_1, x_1)$ defined by

$$\Sigma_1 = \{f_1, f_2, g_1, g_2\}, \qquad D_1 = \{0, 1\} \times \{0, 1\} \times N \times N, \qquad x_1 = \langle 0, 0, 1, 0 \rangle.$$

The partial functions $\bar{f}_i$ and $\bar{g}_i$ are defined as follows:

$$\bar{f}_1(i, j, n, m) = \langle 1, j, n - 1, m \rangle \qquad \text{if} \quad i = 0, n > 0,$$

$$\bar{f}_2(i, j, n, m) = \langle 0, j, n + 1, m + 1 \rangle, \qquad \text{if} \quad i = 1,$$

$$\bar{g}_1(i, j, n, m) = \langle i, 1, n - 1, m - 1 \rangle, \qquad \text{if} \quad j = 0, n > 0, \quad m > 0.$$

$$\bar{g}_2(i, j, n, m) = \langle i, 0, n + 1, m \rangle, \qquad \text{if} \quad j = 1.$$

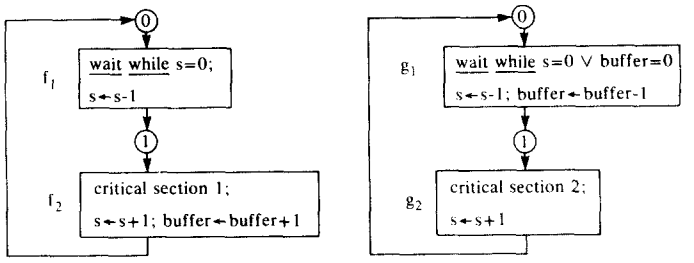


FIG. 3.1. A parallel program.

Let $S_2 = (\Sigma_2, D_2, x_2)$ be the computation system defined by

$$\Sigma_2 = \{a, b\}, \qquad D_2 = N, \qquad x_2 = 0, \qquad \bar{a}(n) = n + 1,$$
$$\bar{b}(n) = n - 1 \qquad \text{if} \quad n > 0.$$

Let ρ: $D_2 \to D_1$ and τ: $\Sigma_2^* \to \Sigma_1^*$ be defined by

$$\rho(n) = \langle 0, 0, 1, n\rangle, \qquad \tau(a) = f_1 f_2, \qquad \tau(b) = g_1 g_2.$$

Then $h = (\rho, \tau)$ is a principal homomorphism. In fact,

$$\iota(\tau(a)) = \{f_1, f_2\}, \qquad \iota(\tau(b)) = \{g_1, g_2\}, \qquad \iota(\tau(a)) \cap \iota(\tau(b)) = \varnothing.$$

Let $\leqslant$ be the partial order on D_1 defined by

$$\langle i, j, n, m\rangle \leqslant \langle i', j', n', m'\rangle \qquad \text{iff} \quad i = i', \quad j = j', \quad n \leqslant n', \quad m \leqslant m'$$

where $\leqslant$ is the usual order on N. Then h is order preserving. Hence S_1 is emulated by S_2 with respect to the order preserving principal homomorphism h.

Let $S_3 = (\Sigma_3, D_3, x_3)$ be the computation system defined by

$$\Sigma_3 = \{a_1, a_2, b_1, b_2\}, \qquad D_3 = N^6, \qquad x_3 = \langle 1, 0, 1, 0, 1, 0\rangle,$$
$$a_1 = \langle -1, 1, 0, 0, -1, 0\rangle, \qquad b_1 = \langle 0, 0, -1, 1, -1, -1\rangle,$$
$$a_2 = \langle 1, -1, 0, 0, 1, 1\rangle, \qquad b_2 = \langle 0, 0, 1, -1, 1, 0\rangle.$$

For each $a = \langle k_1, \ldots, k_6\rangle$ in Σ_3, $\bar{a}$: $D_3 \to D_3$ is defined by

$$\bar{a}(n_1, \ldots, n_6) = \langle n_1 + k_1, \ldots, n_6 + k_6\rangle \qquad \text{if} \quad n_1 + k_1 \geqslant 0, \ldots, n_6 + k_6 \geqslant 0.$$

A computation system of this type is called a *vector addition system* [8], we shall discuss this type of computation system more intensively in a later section. Let ρ': $D_1 \to D_3$ and τ': $\Sigma_1^* \to \Sigma_3^*$ be defined by

$$\tau'(f_k) = a_k, \qquad \tau'(g_k) = b_k, \qquad k = 1, 2,$$
$$\rho'(i, j, n, m) = \langle i, \hat{\imath}, j, \hat{\jmath}, n, m\rangle,$$

where $\hat{0} = 1$, $\hat{1} = 0$. Then $h' = (\rho', \tau')$ is a principal homomorphism. Let $\leqslant$ be the order on N^6 defined by

$$\langle k_1,..., k_6\rangle \leqslant \langle l_1,..., l_6\rangle \qquad \text{iff} \quad k_1 \leqslant l_1,..., k_6 \leqslant l_6.$$

Then h' is order preserving. Hence S_1 is simulated by S_3 with respect to the order preserving principal homomorphism h'.

One of the important problems in parallel processes is the problem called the *mutual exclusion problem.* The mutual exclusion problem for a set of parallel processes is whether, at any instant of time, at most one process is permitted to be in its critical section. Thus, S_1 is mutually exclusive if and only if there is no y in R_{S_1} such that

$$y \geqslant \langle 1, 1, 0, 0\rangle.$$

In the next section, we shall show that this problem, which is expressed in terms of exceedability, can be reduced to the corresponding problem in S_3 by means of the order preserving homomorphism h'.

4. Problems Which Are Preserved by Homomorphisms

In this section, we consider a number of problems which arise in parallel computation. We show that all of these problems are preserved by order preserving principal homomorphisms.

Some problems, however, cannot be expressed directly in terms of computation sequences and reachability sets. We shall not consider such problems, which include mutual exclusion and concurrency, in this paper.

We consider the following problems, assuming $S = (\Sigma, D, x)$ is given:

(1) Reachability: For $y \in D$, is $y \in R_S$?

That is, starting in state x is there a computation sequence of S which ends in state y.

(2) Deadlock: Does there exist an $\alpha \in C_S$ such that, for every $a \in \Sigma$, $\alpha a \notin C_S$.

That is, is it possible to reach a state from which no operation can be performed, causing the whole system to come to a halt.

(3) Termination: Is C_S finite?

That is, is it true that, no matter how the system runs, each operation is performed only a finite number of times.

(4) Finiteness: Is R_S finite?

That is, is there only a finite number of states that can be reached in the system starting from the initial state x?

(5) Equivalence of sets of computation sequences: For $y, z \in D$, is $C_S(y) = C_S(z)$?

That is, is α a computation sequence starting with state y if and only if α is also a computation sequence starting with state z? Note that it follows from the definitions for computation sequences that $C_S(y) = C_S(z)$ iff $C_S^\omega(y) = C_S^\omega(z)$.

(6) Liveness: For any $\alpha \in C_S$ and $a \in \Sigma$, does there exist a $\beta \in \Sigma^*$ such that $\alpha\beta a \in C_S$?

That is, is the system such that, no matter what state is reached for any operation a, it is still possible to continue the computation sequence such that a can be performed again.

(7) Exceedability: With D a partially ordered set and given $y \in D$, does there exist a $z \in R_S$ such that $z \geqslant y$?

These properties have been previously studied in particular models. Reachability, finiteness, and exceedability have been studied particularly in vector addition systems. The liveness problem has been studied extensively in the Petri net literature, and deadlock is of interest in cooperating sequential process. The property of lockout or starvation has also been of considerable interest. Since lockout seems to involve several concepts which are not required for the other properties, and there seem to be several different forms of lockout, we do not include a discussion of lockout in this paper.

DEFINITION 4.1. Let H be a class of homomorphisms. Let P be a problem of the form: Given a computation system $S = (\Sigma, D, x)$, $y_1, \ldots, y_n \in D$, whether $P(S, y_1, \ldots, y_n)$? We say that P *is preserved by* H under the following condition: For any S_1 and S_2, if there is an $h \in H$ such that $h: S_1 \to S_2$, then

$$P(S_1, y_1, \ldots, y_n) \quad \text{holds} \quad \text{iff} \quad P(S_2, \rho(y_1), \ldots, \rho(y_n)) \quad \text{holds.}$$

This notion of a problem being preserved by H provides a basic mechanism for transferring knowledge of properties from one class of systems to another.

Let C_1 and C_2 be classes of computation systems. Suppose that C_1 is simulated by C_2 with respect to H. Let P be a problem preserved by H; then the problem P in C_1 is recursively reducible to the problem P in C_2; that is, if P is decidable in C_2, then P is also decidable in C_1. Similarly, if P is undecidable in C_1, then P is also undecidable in C_2. If P in C_1 is recursively reducible to P in C_2 and P in C_2 is recursively reducible to P in C_1, then P in C_1 is recursively equivalent to P in C_2, that is, one is decidable iff the other is.

THEOREM 4.1. *A spanning homomorphism preserves reachability.*

Proof. Let $h = (\rho, \tau)$ be a spanning homomorphism from S_1 to S_2. Then $y \in R_{S_1}$:

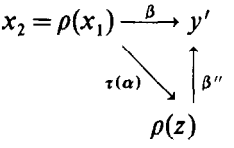
iff $x_1 \xrightarrow{\alpha} y$ for some $\alpha \in \Sigma_1^*$,

iff $\rho(x_1) \xrightarrow{\beta} \rho(y)$ for some $\beta \in \Sigma_2^*$, by (2.2), (2.5),

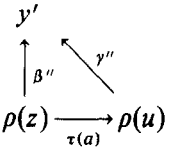
iff $\rho(y) \in R_{S_2}$ by (2.1). ∎

THEOREM 4.2. *A spanning homomorphism preserves deadlock.*

Proof. Let h: $S_1 \to S_2$ be a spanning homomorphism. Suppose that S_2 has a deadlock. Then there exists a $\beta \in C_{S_2}$ such that $\beta b \notin C_{S_2}$ for all $b \in \Sigma_2$. Let y' be the element such that $x_2 \to^{\beta} y'$. Then $\bar{b}(y') = \perp$ for all $b \in \Sigma_2$. Let k be an integer which satisfies condition (2.6). Then, by (2.6), there exist $a \in \Sigma_1^*$, $z \in D_1$, $\beta'' \in \Sigma_2^*$ such that $|\beta''| \leqslant k$ and



Choose z and β'' such that the length of β'' is minimum. Suppose that $\bar{a}(z) \neq \perp$ for some $a \in \Sigma_1$. Then by (2.6), there exist $u \in D_1$ and $\gamma'' \in \Sigma_2^*$ such that $\beta'' \simeq \tau(a)\gamma''$ and

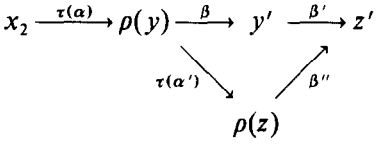


Since $|\gamma''| < |\beta''|$, this contradicts the minimality of $|\beta''|$. Hence $\bar{a}(z) = \perp$ for all $a \in \Sigma$. Thus, S_1 has a deadlock.

Suppose that S_1 has a deadlock. Then there exists an $\alpha \in C_{S_1}$ such that $\alpha a \notin C_{S_1}$ for all $a \in \Sigma_1$. Let y be the element such that $x_1 \to^{\alpha} y$. Suppose that there is a computation of S_2 such that

$$\rho(y) \xrightarrow{\beta} y', \qquad |\beta| > k.$$

Then, by (2.6), there exist $z \in D_1$, $a' \in \Sigma_1^*$, $z' \in D_2$, β', $\beta'' \in \Sigma_2^*$ such that $|\beta''| \leqslant k$ and



Since $|\beta''| \leqslant k$ and $|\tau(\alpha')\beta''| = |\beta\beta'| \geqslant |\beta|$, $\tau(\alpha') \neq \lambda$. Hence $y \to \alpha' z$ for some $\alpha' \in \Sigma_1^* - \{\lambda\}$. This contradicts the fact $\bar{a}(y) = \perp$ for all $a \in \Sigma_1$. Hence $\rho(y) \to^{\beta} y'$ implies $|\beta| \leqslant k$. Let $\rho(y) \to^{\beta} y'$ be one of the longest computations from $\rho(y)$. Then $\bar{b}(y') = \perp$ for all $b \in \Sigma_2$. Hence S_2 has a deadlock. ∎

THEOREM 4.3. *A spanning homomorphism preserves the termination property.*

Proof. Let h: $S_1 \to S_2$ be a spanning homomorphism. Let k be an integer which satisfies (2.6). Suppose that C_{S_1} is infinite. Then for any $n \in N$, there is a computation of S_1 such that

$$x_1 \xrightarrow{\beta} y, \qquad |\beta| > n.$$

Hence, for any n, there is a computation of S_2 such that

$$x_2 \xrightarrow{\tau(\beta)} \rho(y), \qquad |\tau(\beta)| > n.$$

Hence C_{S_2} is infinite.

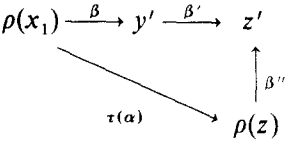
Suppose that C_{S_2} is infinite. Let

$$m = \max\{|\tau(a)| \mid a \in \Sigma_1\}.$$

Then, for any $n \in N$, there is a computation of S_2 such that

$$x_2 = \rho(x_1) \xrightarrow{\beta} y', \qquad |\beta| > m \cdot n + k.$$

By (2.6), there exist $\alpha \in \Sigma_1^*$, $\beta', \beta'' \in \Sigma_2^*$, $z' \in D_2$, $z \in D_1$ such that $|\beta\beta'| = |\tau(\alpha)\beta''|$, $|\beta''| \leqslant k$, and



Since $|\beta''| \leqslant k$,

$$|\tau(\alpha)| \geqslant |\beta| + |\beta'| - k > m \cdot n.$$

By (2.5), there exists $\alpha' \in \Sigma_1^*$ such that

$$x_1 \xrightarrow{\alpha'} z \qquad \text{and} \qquad \tau(\alpha') \simeq \tau(\alpha).$$

Since $|\tau(\alpha')| = |\tau(\alpha)| > m \cdot n$, $|\alpha'| > n$. Therefore, for any $n \in N$, there is a computation of S_1 such that

$$x_1 \xrightarrow{\alpha'} z \qquad \text{and} \qquad |\alpha'| > n.$$

Hence C_{S_1} is infinite. ∎

THEOREM 4.4. *A surjective homomorphism preserves finiteness.*

Proof. Let h: $S_1 \to S_2$ be a surjective homomorphism. By Lemma 2.1, $\rho(R_{S_1}) \subseteq R_{S_2}$. Since ρ is injective, $\#R_{S_1} \leqslant \#R_{S_2}$. Let $n = \#\Sigma_2$ and $m = \max_{a \in \Sigma_1} |\tau(a)|$. For each $y' \in R_{S_2}$, let $V(y')$ be the subset of D_2 defined by

$$V(y') = \{\bar{\beta}(y') \mid \bar{\beta}(y') \neq \perp, \beta \in \Sigma_2^*, |\beta| \leqslant m\}.$$

That is, $z' \in V(y')$ if and only if there exists a path from y' to z' of length less than $m + 1$. Then, from (2.3) and (2.4), it follows that

$$R_{S_2} = \bigcup_{y \in R_{S_1}} V(\rho(y)).$$

Since, $\#V(y') \leqslant n^m$ for each $y' \in R_{S_2}$,

$$\#R_{S_2} \leqslant n^m \cdot \#R_{S_1}.$$

Hence R_{S_1} is finite if and only if R_{S_2} is finite. ∎

We shall see later that a spanning homomorphism does not preserve finiteness.

THEOREM 4.5. *A principal homomorphism preserves equivalence of sets of computation sequences.*

Proof. Let h: $S_1 \to S_2$ be a principal homomorphism. Let y and z be in R_{S_1}. Suppose that $C_{S_2}(\rho(y)) = C_{S_2}(\rho(z))$. Let α be an arbitrary element of $C_{S_1}(y)$. Then

$$y \xrightarrow{\alpha} y' \quad \text{in} \quad S_1 \quad \text{for some} \quad y'.$$

Then, by (2.2), $\rho(y) \to^{\tau(\alpha)} \rho(y')$. Hence, $\tau(\alpha) \in C_{S_2}(\rho(y))$, so $\tau(\alpha) \in C_{S_2}(\rho(z))$. Then there exists a $w \in R_{S_2}$ such that

$$\rho(z) \xrightarrow{\tau(\alpha)} w.$$

By Corollary 3.2, there exists $z' \in R_{S_1}$ such that

$$z \xrightarrow{\alpha} z', \qquad \rho(z') = w.$$

Therefore $\alpha \in C_{S_1}(z)$. Hence we have $C_{S_1}(y) \subseteq C_{S_1}(z)$. By symmetry, $C_{S_1}(z) \subseteq C_{S_1}(y)$. Hence we have $C_{S_1}(y) = C_{S_1}(z)$.

Suppose that $C_{S_1}(y) = C_{S_1}(z)$. Let β be an arbitrary element of $C_{S_2}(\rho(y))$. Then

$$\rho(y) \xrightarrow{\beta} y', \qquad \text{for some} \quad y' \in R_{S_2}.$$

Then, by (2.4), there exist $\beta' \in \Sigma_2^*$, $y'' \in R_{S_1}$ such that

$$y' \xrightarrow{\beta'} \rho(y'').$$

By (2.3), there exists $\alpha \in \Sigma_1^*$ such that

$$y \xrightarrow{\alpha} y'', \qquad \tau(\alpha) = \beta\beta'.$$

Since $\alpha \in C_{S_1}(y) = C_{S_1}(z)$, there exists $z'' \in R_{S_1}$ such that

$$z \xrightarrow{\alpha} z''.$$

Hence, by (2.2),

$$\rho(z) \xrightarrow{\beta\beta'} \rho(z'').$$

Thus $\beta \in C_{S_2}(\rho(z))$, so $C_{S_2}(\rho(y)) \subseteq C_{S_2}(\rho(z))$. By symmetry, $C_{S_2}(\rho(z)) \subseteq C_{S_2}(\rho(y))$. Hence we have $C_{S_2}(\rho(y)) = C_{S_2}(\rho(z))$. ∎

LEMMA 4.1. *Let h: $S_1 \to S_2$ be a spanning homomorphism such that there exists a function f: $\Sigma_1 \to \Sigma_2$ which statisfies the following*:

(i) *for $a, b \in \Sigma_1$, $\bar{a} \neq \bar{b}$ implies $f(a) \neq f(b)$,*

(ii) *for each $a \in \Sigma_1$, $\tau(a) = \gamma f(a)\gamma_2$, $\gamma_1, \gamma_2 \in (\Sigma_2 - f(\Sigma_1))^*$,*

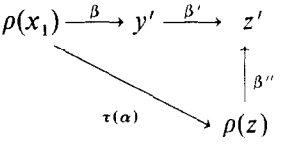
(iii) $\Sigma_2 = \bigcup_{a \in \Sigma_1} \iota(\tau(a))$.

Then, S_1 is live if and only if S_2 is live.

Proof. Let k be an integer which satisfies (2.6). Suppose that S_1 is live. Let β be any element of C_{S_2}. Then,

$$\rho(x_1) = x_2 \xrightarrow{\beta} y' \qquad \text{for some} \quad y' \in D_2.$$

By (2.6), there exist, α, z, z', β', β'' such that $|\beta''| \leqslant k$ and



Let b be any element of Σ_2. Then, by (iii), there exist $a \in \Sigma_1$ and $\gamma_1, \gamma_2 \in \Sigma_2^*$ such that

$$\tau(a) = \gamma_1 b \gamma_2.$$

Since S_1 is live, there exist $\gamma \in \Sigma_1^*$ and $u \in R_{S_1}$ such that

$$z \xrightarrow{\gamma} u$$

and γ contains more than k occurrences of a. Then, by (2.6), there exist u', γ', γ'' such that

$$\begin{array}{ccc} z' & \xrightarrow{\gamma'} & u' \\ \uparrow{\scriptstyle \beta''} & & \uparrow{\scriptstyle \gamma''} \\ \rho(z) & \xrightarrow[\tau(\gamma)]{} & \rho(u) \end{array}$$

and $\tau(\gamma)\gamma'' \simeq \beta''\gamma'$. Since $|\beta''| \leqslant k$ and γ contains more than k occurrences of a, γ' contains at least one occurrence of b. Since $\rho(x_1) \to^{\beta} y' \to^{\beta'\gamma'} u'$ and γ' contains b, S_2 is live.

Suppose that S_2 is live. Let α be any element of C_{S_1} and a be any element of Σ_1. Then

$$x_1 \xrightarrow{\alpha} y \qquad \text{for some} \quad y.$$

Then, by (2.2)

$$\rho(x_1) \xrightarrow{\tau(\alpha)} \rho(y).$$

Let $\tau(a) = b\gamma_1$, $b \in \Sigma_2$, $\gamma_1 \in \Sigma_2^*$. Since S_2 is live, there exist $\beta \in \Sigma_2^*$, $y' \in D_2$ such that

$$\rho(y) \xrightarrow{\beta} y'$$

and β contains more than k occurrences of b. Then, by (2.6), there exist α', z, z', β', β'' such that

$$\begin{array}{ccccc} \rho(y) & \xrightarrow{\beta} & y' & \xrightarrow{\beta'} & z' \\ & \searrow^{\tau(\alpha')} & & & \uparrow{\scriptstyle \beta''} \\ & & & & \rho(z) \end{array}$$

and $\tau(\alpha')\beta'' \simeq \beta\beta'$, $|\beta''| \leqslant k$. Hence

$$y \xrightarrow{\alpha'} z,$$

and $\tau(\alpha')$ contains $f(a)$. Then there exist α_1, $\alpha_2 \in \Sigma_1^*$ and $a' \in \Sigma_1$ such that $\alpha' = \alpha_1 a' \alpha_2$ and $\tau(a')$ contains $f(a)$. By (ii), $f(a) = f(a')$. By (i), $\bar{a} = \bar{a}'$. Thus,

$$y \xrightarrow{\alpha_1 a \alpha_2} z.$$

Therefore S_1 is live. ∎

THEOREM 4.6. *A principal homomorphism preserves liveness.*

Proof. Let h: $S_1 \to S_2$ be a principal homomorphism. Let f: $\Sigma_1 \to \Sigma_2$ be defined by

$$f(a) = b \qquad \text{if} \quad \tau(a) = b\gamma \quad \text{for some} \quad \gamma \in \Sigma_2^*.$$

Then f satisfies conditions (i)–(iii) of Lemma 4.1. ∎

THEOREM 4.7. *An order preserving spanning homomorphism preserves exceedability.*

Proof. Let h: $S_1 \to S_2$ be an order preserving spanning homomorphism. Let y be an element of D_1. Suppose that there exists a $z \in R_{S_1}$ such that $z \geqslant y$. By Theorem 4.1, $\rho(z) \in R_{S_2}$. Hence by (3.2), $\rho(y) \geqslant \rho(z)$.

Suppose that there exists a $z' \in R_{S_2}$ such that $z' \geqslant \rho(y)$. Since $\rho(y) \in \rho(D_1)$, it follows from (3.3) that $z' \in \rho(D_1)$, that is, there exists a $z \in D_1$ such that $z' = \rho(z)$. By Theorem 4.1, $z \in R_{S_1}$. Hence by (3.2), $z \geqslant y$. ∎

It should be immediately evident from the definitions of the various properties given at the start of this section that there are relationships between them. For example, termination implies deadlock and deadlock implies nonlinveness. The converses of these implications do not, in general, hold. Later we shall show some examples demonstrating these relationships.

In the following sections, we shall show how certain models of parallel computation can be simulated by other models of parallel computation under the types of homomorphisms we have defined; i.e., by spanning homomorphisms, principal homomorphisms, or isomorphisms. Then, using the results in this section which preserve properties under these homomorphisms, it directly follows that the properties that hold in one model can be determined via the properties of the simulating model.

5. VECTOR ADDITION SYSTEMS, VECTOR REPLACEMENT SYSTEMS, AND GENERALIZED PETRI NETS

In this section, we wish to show relationships between vector addition systems, vector replacement systems, and generalized Petri nets. As before, we let Z denote the set of integers and N the set of natural numbers. For each $x \in Z^n$, and i, $1 \leqslant i \leqslant n$, $x(i)$ denotes the ith component of x. For each x and y in Z^n, the sum $x + y$ is defined by

$$(x + y)(i) = x(i) + y(i), \qquad 1 \leqslant i \leqslant n.$$

For $x, y \in Z^n$, we write $x \geqslant y$ if $x(i) \geqslant y(i)$ for all i, $1 \leqslant i \leqslant n$. Note that $\geqslant$ is a partial order on Z^n. We denote the zero vector $\langle 0,..., 0\rangle$ by 0^n.

DEFINITION 5.1. An *n-dimensional vector addition system* is a pair $V = (\Sigma, x)$ in which x is an n-dimensional vector of nonnegative integers, and Σ is a finite set of n-

dimensional integer vectors. Let $D = N^n$. For each $a \in \Sigma$, the partial function $\bar{a}$: $D \to D$ is defined as follows:

$$\begin{aligned} \bar{a}(y) &= y + a, && \text{if} \quad y + a \geqslant 0, \\ &= \perp, && \text{otherwise.} \end{aligned}$$

The computation system $S_V = (\Sigma, D, x)$ is said to be *realized by* $V = (\Sigma, x)$.

DEFINITION 5.2. An *n-dimensional vector replacement system* $R = (\Sigma, x)$ consists of:

(i) x, an n-dimensional vector of nonnegative integers, and

(ii) Σ, a fnite set of ordered pairs of n-dimensional integer vectors

$$\Sigma = \{(u_1, v_1), (u_2, v_2), \ldots, (u_p, v_p)\},$$

where $u_i \leqslant 0$ and $u_i \leqslant v_i$, $i = 1, 2, \ldots, p$.

The u_i vectors are called *test vectors* and the v_i are called *replacement vectors*. Let $D = N^n$. For each $a = (u, v)$ in Σ, the partial function $\bar{a}$: $D \to D$ is defined as

$$\begin{aligned} \bar{a}(y) &= y + v, && \text{if} \quad y + u \geqslant 0, \\ &= \perp, && \text{otherwise.} \end{aligned}$$

$S_R = (\Sigma, D, x)$ is said to be the computation system *realized by* R.

We denote, by VAS and VRS, the classes of computation systems realized by vector addition systems and vector replacement systems, respectively.

THEOREM 5.1. *Every* VRS *is simulated by a* VAS *with respect to order preserving principal homomorphism.*

Proof. Let $R = (\Sigma_1, x_1)$ be a n-dimensional vector replacement system. Let $m = \#\Sigma_1$. Now we construct an $(n + m + 1)$-dimensional vector addition system $V = (\Sigma_2, x_2)$. Let $\Sigma_1 = \{a_1, a_2, \ldots, a_m\}$. For each $a_i = (u_i, v_i)$, let a_i' and a_i'' be elements of Z^{n+m+1} defined as follows:

$$\begin{aligned} a_i'(j) &= u_i(j), && \text{for} \quad 1 \leqslant j \leqslant n, \\ &= 1, && \text{for} \quad j = n + i, \\ &= -1, && \text{for} \quad j = n + m + 1, \\ &= 0, && \text{otherwise;} \end{aligned}$$

$$\begin{aligned} a_i''(j) &= v_i(j) - u_i(j), && \text{for} \quad 1 \leqslant j \leqslant n, \\ &= -1, && \text{for} \quad j = n + i, \\ &= 1, && \text{for} \quad j = n + m + 1, \\ &= 0, && \text{otherwise.} \end{aligned}$$

Let $\Sigma_2 = \{a_1', a_1'', ..., a_m', a_m''\}$. The initial element x_2 of V is defined by

$$\begin{aligned} x_2(j) &= x_1(j), && \text{for} \quad 1 \leqslant j \leqslant n, \\ &= 0, && \text{for} \quad n < j \leqslant n+m, \\ &= 1, && \text{for} \quad j = n+m+1. \end{aligned}$$

Now we construct a homomorphism h: $S_R \to S_V$. Let τ: $\Sigma_1^* \to \Sigma_2^*$ and ρ: $N^n \to N^{n+m+1}$ be defined as

$$\tau(a_i) = a_i' a_i'' \qquad \text{for} \quad 1 \leqslant i \leqslant m.$$

$$\begin{aligned} \rho(x)(j) &= x(j), && \text{for} \quad 1 \leqslant j \leqslant n, \\ &= 0, && \text{for} \quad n < j \leqslant n+m, \\ &= 1, && \text{for} \quad j = n+m+1. \end{aligned}$$

Now we show that $h = (\tau, \rho)$ is an order preserving principal homomorphism from S_R to S_V. Clearly, (2.1) is satisfied. Suppose that

$$y \xrightarrow{a_i} z \qquad \text{in} \quad S_R.$$

Then $y(j) + u_i(j) \geqslant 0$ and $z(j) = y(j)$, $1 \leqslant j \leqslant n$. Thus

$$\begin{aligned} \rho(y) &= \langle y, \overbrace{0, 0, ..., 0}^{m}, 1 \rangle \\ &\xrightarrow{a_i'} \langle s, \overbrace{0, ..., 1}^{i}, 0, ..., 0 \rangle = s' \\ &\xrightarrow{a_i''} \langle t, 0, 0, ..., 0, 1 \rangle = \rho(t), \end{aligned}$$

where for $j = 1, ..., n$,

$$\begin{aligned} s(j) &= y(j) + u_i(j), \\ t(j) &= s(j) + (v_i(j) - u_i(j)) = y(j) + v_i(j) = z(j). \end{aligned}$$

Hence $z = t$, $\rho(y) \to^{a_i' a_i''} \rho(z)$. Therefore, (2.2) holds, so h is a homomorphism.

Suppose that $\rho(y) \to^b w'$ in S_V for some $y \in N^n$, $w' \in N^{n+m+1}$, $b \in \Sigma_2$. Then $b = a_i'$ for some i. (Since $\rho(y)(j) = 0$ for $j = n+1, ..., n+m$, a_i'' cannot be applied to $\rho(y)$ for any i.) Since $w'(n+i) = 1$ and $w'(n+m+1) = 0$, the only applicable operation to w' is a_i''. Then

$$\rho(y) \xrightarrow{a_i'} w' \xrightarrow{a_i''} \rho(z) \qquad \text{is in} \quad S_2 \qquad \text{and} \qquad y \xrightarrow{a_i} z \qquad \text{is in} \quad S_1.$$

Hence (2.3) and (2.4) hold. Therefore h is surjective. Since $\iota(\tau(a_i)) \cap \iota(\tau(a_j)) = \varnothing$, $i \neq j$, h is principal.

Clearly (3.2) holds. Suppose that $x_2 \to^* y$ is in S_V. Then there exists an i, $1 \leqslant i \leqslant n+1$ such that

$$y(n+i) = 1 \qquad \text{and} \qquad y(n+j) = 0 \qquad \text{for} \quad j \neq i, \quad 1 \leqslant j \leqslant m+1.$$

Then $y \in \rho(N^n)$ if and only if $y(n+m+1) = 1$ and $y(j) = 0$ for $j = n+1, \ldots, n+m$. Thus (3.3) holds. Hence h is order preserving. ∎

Remark. In [9] it was shown that the exceedability and termination problems are decidable in VAS. Thus, using the results of Section 4, these problems are also decidable in VRS. This yields another proof of this result given in [10].

EXAMPLE 5.1. Consider the 2-dimensional vector replacement system $R = (\{a, b\}, \langle 1, 0\rangle)$, where

$$a = (\langle -1, 0\rangle, \langle -1, 1\rangle) \qquad \text{and} \qquad b = (\langle 0, -1\rangle, \langle 1, 0\rangle).$$

Following the construction in the proof of Theorem 5.1 gives a 5-dimensional vector addition system defined by $V = (\Sigma, \langle 1, 0, 0, 0, 1\rangle)$, where Σ consists of

$$a' = \langle -1, 0, 1, 0, -1\rangle, \qquad a'' = \langle 0, 1, -1, 0, 1\rangle,$$
$$b' = \langle 0, -1, 0, 1, -1\rangle, \qquad b'' = \langle 1, 1, 0, -1, 1\rangle,$$

Here the vectors a' and a'' simulate a and the vectors b' and b'' simulate b.

DEFINITION 5.3. A *generalized Petri net* $P = (\Pi, \Sigma, \Delta, x)$ consists of:

(i) a finite set Π called *places*,

(ii) a finite set Σ called *transitions*, $\Pi \cap \Sigma = \emptyset$,

(iii) an *incidence function* Δ: $\Pi \times \Sigma \cup \Sigma \times \Pi \to N$,

(iv) an *initial marking* x: $\Pi \to N$.

A function y: $\Pi \to N$ is called a *marking*. When $\Pi = \{p_1, p_2, \ldots, p_n\}$, we sometimes regard a marking y as an n-dimensional vector $\langle y(p_1), y(p_2), \ldots, y(p_n)\rangle$. Let D_P be the set of markings of P. For each $a \in \Sigma$, a partial function $\bar{a}$: $D_P \to D_P$ is defined as follows: Let $y \in D_P$. Then $\bar{a}(y)$ is defined if and only if $y(p_i) \geqslant \Delta(p_i, a)$ for all $p_i \in \Pi$. Suppose that $y(p_i) \geqslant \Delta(p_i, a)$ for all $p_i \in \Pi$. Then

$$\bar{a}(y)(p_i) = y(p_i) - \Delta(p_i, a) + \Delta(a, p_i), \qquad p_i \in \Pi.$$

The computation system $S_P = (\Sigma, D_P, x)$ is said to be *realized by* P.

We denote by GPN the class of computation systems realized by generalized Petri nets. A *Petri net* is a generalized Petri net in which for each $p_i \in \Pi$ and $a \in \Sigma$,

$$0 \leqslant \Delta(p_i, a) \leqslant 1, \qquad 0 \leqslant \Delta(a, p_i) \leqslant 1.$$

A generalized Petri net P is drawn as a graph in which each place is usually represented by a circle and each transition by a bar. A place p is connected to a transition a by an arc labeled with the integer $\Delta(p, a)$ if $\Delta(p, a) \neq 0$, and a transition a is connected to a place p by an arc with $\Delta(a, p)$ if $\Delta(a, p) \neq 0$. Places are used to hold markers called *tokens*. For a given transition a, we call those places p such that $\Delta(p, a) > 0$, *input places* of a; similarly, the places such that $\Delta(a, p) > 0$ are called *output places* of a. For the sake of clarity, arcs of size 1 are usually not labeled. The initial marking is shown by denoting single tokens by dots in places.

A transition a is enabled if its input places contain enough tokens. The firing of the transition produces a new marking by removing tokens from the input places and adding tokens to the output places as indicated by the size of the arcs.

A computation sequence for a Petri net is usually called a *firing sequence*. A transition a is said to be *fireable* at a marking y if $\bar{a}(y) \neq \perp$.

THEOREM 5.2. *The class* GPN *can be simulated by* VRS *with respect to length preserving principal homomorphisms.*

Proof. Let $P = (\Pi, \Sigma_1, \Delta, x)$ be a generalized Petri net, for which $\Pi = \{p_1, p_2, ..., p_n\}$ and $\Sigma_1 = \{a_1, a_2, ..., a_m\}$. We now construct an n-dimensional vector replacement system $R = (\Sigma_2, N^n, x')$ as follows:

(i) $x' = \langle x(p_1), x(p_2), ..., x(p_n)\rangle$,

(ii) for each $a_i \in \Sigma_1$, let a_i' be the pair of vectors (u_i, v_i) defined by

$$u_i(j) = -\Delta(p_j, a_i), \qquad j = 1, ..., n,$$
$$v_i(j) = \Delta(a_i, p_j) - \Delta(p_j, a_i), \qquad j = 1, ..., n.$$

Let $\Sigma_2 = \{a_i' \mid 1 \leqslant i \leqslant m\}$. Let $\tau\colon \Sigma_1^* \to \Sigma_2^*$ and $\rho\colon D_P \to N^n$ be the function defined as follows:

$$\tau(a_i) = a_i', \qquad i = 1, 2, ..., m, \qquad \rho(y)(j) = x(p_j), \qquad y \in D_P, \quad 1 \leqslant j \leqslant n.$$

Then, $\rho\colon D_P \to N^n$ is bijective. Clearly h satisfies (2.1) and (3.1). Thus h is a length preserving principal homomorphism. ∎

EXAMPLE 5.2. $S_R = (\{a_1', a_2', a_3'\}, N^2, \langle 1, 0\rangle)$, $a_1' = a_2' = (\langle -1, 0\rangle, \langle -1, 1\rangle)$, $a_3' = (\langle 0, -1\rangle, \langle 1, 0\rangle)$. (See Fig. 5.1.)

DEFINITION 5.4. Two transitions a and b are said to be *equivalent* if $\bar{a} = \bar{b}$, that is, for all $i \in \Pi$,

$$\Delta(i, a) = \Delta(i, b) \qquad \text{and} \qquad \Delta(a, i) = \Delta(b, i).$$

A generalized Petri net is called *irreflexive* if and only if there does not exist any $i \in \Pi$ and $a \in \Sigma$ such that $\Delta(a, i) > 0$ and $\Delta(i, a) > 0$.

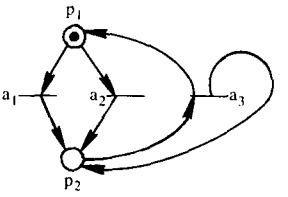


FIG. 5.1. A Petri net and the corresponding vector replacement system.

These equivalence and irreflexive properties can be found in [16]. Also, see [10].

COROLLARY 5.1. *The class of generalized Petri nets without equivalent transitions and the class* VRS *can simulate each other with respect to isomorphisms.*

Proof. Let P be a generalized Petri net without equivalent transitions. Let R be the vector replacement system and h: $S_P \to S_R$ be the homomorphism constructed as in Theorem 5.2. Then for any distinct operations a_i and a_j in Σ_1, $\tau(a_i) \neq \tau(a_j)$. Thus τ: $\Sigma_1^* \to \Sigma_2^*$ is bijective. Note that ρ is bijective and h satisfies (2.1) and (3.1). Hence, by Corollary 3.1, h is an isomorphism.

Suppose that a vector replacement system $R = (\Sigma_2, N^n, x)$ is given. Then, we can construct a generalized Petri net without equivalent transitions, using the same correspondence as in Theorem 5.2. Thus, VRS can be simulated by generalized Petri nets without equivalent transitions with respect to isomorphisms. ∎

COROLLARY 5.2. *The class of irreflexive generalized Petri nets without equivalent transitions and the class* VAS *can simulate each other with respect to isomorphisms.*

Proof. Let P be an irreflexive generalized Petri net without equivalent transitions. Let R be the vector replacement system constructed from P as in Theorem 5.2. Now a vector addition system can be regarded as a special type of vector replacement system for which $u_i(j) = \min\{0, v_i(j)\}$ for all i and j. The proof of equivalence between the VAS and the VRS satisfying this condition should be evident. The system constructed from P satisfies this condition, so R can be regarded as a vector addition system. Since P contains no equivalent transitions, h is an isomorphism.

Conversely, given a vector addition system R we can construct an isomorphic generalized Petri net P, using the same correspondence as in Theorem 5.2. ∎

Our correspondences between generalized Petri nets, vector addition systems and vector replacement systems are summarized in Fig. 5.2.

Corollaries 5.1 and 5.2 were stated as Propositions 7 and 8, respectively, in [16], but were not formally proved there. Similar relationships have been given previously by Hack, Keller [10], and others. In addition to the formal proofs of these results using our homomorphisms definitions which we supply here, Theorem 5.2 gives a more exact correspondence between GPN and VRS than has appeared previously.

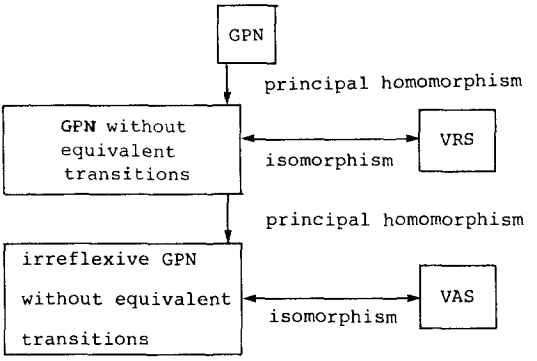


FIG. 5.2. Relationships between GPN, VRS, and VAS.

To conclude this section, we return to the relationships between the various problems mentioned earlier in Section 4. We consider the two Petri nets shown in Fig. 5.3.

Petri net P_1 does not terminate but has a deadlock, showing that deadlock does not imply termination. The point is that an infinite firing sequence is possible—if we assume the rightmost transition never fires. On the other hand, if this transition does fire then the system is deadlocked. Example P_2 has no deadlock but is not live. This can be seen by observing that after the first firing of the rightmost transition, firings of the top loop can no longer occur. Thus nonliveness does not imply deadlock.

Next, we show that a spanning homomorphism does not preserve some properties. Consider Petri nets P_3 and P_4 in Fig. 5.4. Let $h = (\rho, \tau)$ be defined by

$$\rho(n) = <\langle n, 0\rangle, \qquad n \geqslant 0, \qquad \tau(a) = bb'.$$

Then h is a spanning homomorphism from P_3 to P_4. Note that the reachability set of P_3 is finite, but the reachability set of P_4 is infinite. Hence, in general, a spanning homomorphism does not preserve finiteness.

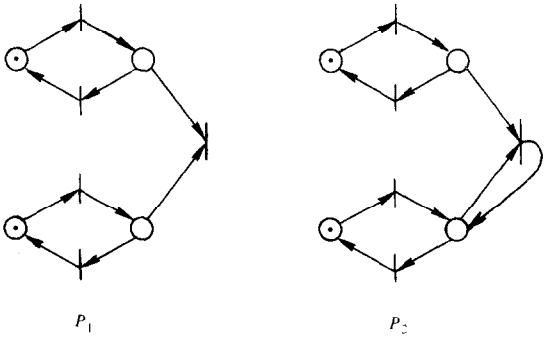


FIG. 5.3. Examples for problem relationships.

FIG. 5.4. An example for a property which is not preserved by a spanning homomorphism.

6. COMPUTATION GRAPHS, UNSHARED PRODUCER-CONSUMER SYSTEMS, AND GENERALIZED MARKED GRAPHS

In this section, we consider another class of parallel computation models. Almost all results stated in this section, as well as those in Section 4, appeared in a less formal setting in an earlier work of one of the authors [16], however the more precise use of homomorphisms used here clarifies a number of issues left open previously.

DEFINITION 6.1. Let $P = (\Pi, \Sigma, \Delta, x)$ be a generalized Petri net. For each $p \in \Pi$, let indeg(p) and outdeg(p) be the number of edges incident into and incident out of p, respectively, that is,

$$\text{indeg}(p) = \#\{a \mid \Delta(a, p) \neq 0\}, \qquad \text{outdeg}(p) = \#\{a \mid \Delta(p, a) \neq 0\}.$$

A *generalized marked graph* is a generalized Petri net such that, for all $p \in \Pi$,

$$\text{indeg}(p) = 1 \qquad \text{and} \qquad \text{outdeg}(p) = 1.$$

A *marked graph* is a generalized marked graph $P = (\Pi, \Sigma, \Delta, x)$ in which, for each $p \in \Pi$ and $a \in \Sigma$,

$$\Delta(p, a) \leqslant 1, \qquad \Delta(a, p) \leqslant 1.$$

The concept of semaphores was introduced by Dijkstra [4] to provide a means of coordinating cooperating sequential process. A semaphore s is a nonnegative integer-valued variable which can be accessed in program processes only by two types of instructions $P(s)$ and $V(s)$, which are defined as follows:

$P(s)$ is an indivisible operation on a semaphore s, and $P(s)$ at location L has the same meaning as

$$L: \textbf{if } s < 1 \textbf{ go to } L \textbf{ else } s \leftarrow s - 1.$$

Of course, if one performance of $P(s)$ is taken and $s < 1$ with a return to L, then this next performance of $P(s)$ must competively seek access to s along with all other operations on s.

An indivisible operation $V(s)$ on a semaphore s, has the same meaning as

$$s \leftarrow s + 1.$$

DEFINITION 6.2. Let $X = \{s_1, s_2, ..., s_n\}$ be a finite set. An element of X is called a *semaphore*. For each i, $1 \leqslant i \leqslant n$, let $P(s_i)$ and $V(s_i)$ be abstract symbols. Let

$$P = \{P(s_i) \mid 1 \leqslant i \leqslant n\}, \qquad V = \{V(s_i) \mid 1 \leqslant i \leqslant n\}.$$

For each i, $1 \leqslant i \leqslant n$, the partial functions $\overline{P(s_i)}$ and $\overline{V(s_i)}$ from N^n to N^n are defined as follows: Let $y = \langle y_1, ..., y_n \rangle \in N^n$. Then $\overline{V(s_i)}(y) = \langle z_1, ..., z_n \rangle$ is defined by

$$\begin{aligned} z_j &= y_i + 1, && \text{if } j = i, \\ &= y_j, && \text{if } j \neq i. \end{aligned}$$

Partial function $\overline{P(s_i)}(y)$ is defined if and only if $y_i \geqslant 1$. In this case $\overline{P(s_i)} = \langle z_1, ..., z_n \rangle$ is defined by

$$\begin{aligned} z_j &= y_i - 1, && \text{if } j = i, \\ &= y_i, && \text{if } j \neq i. \end{aligned}$$

An element of $P^*V^* - \{\lambda\}$ is called a *process* over semaphores $\{s_1, ..., s_n\}$. A *producer–consumer* system is a pair $S = (\Sigma, x)$, where Σ is a finite set of processes and x is an element of N^n. Let a be a process and s be a semaphore. Then, a is called a *producer of* s if a contains a $V(s)$ operation, and s is called a *consumer of* s if a contains a $P(s)$ *operation*. A producer–consumer system S is said to be *unshared* if each semaphore appearing in S has at most one producer and at most one consumer. Let $S = (\Sigma, x)$ be a producer–consumer system. Then for each process $a = P(s_{i_1}) \cdots P(s_{i_k}) V(s_{j_1}) \cdots V(s_{j_l})$ in Σ, the partial function $\bar{a}: N^n \to N^n$ is defined by

$$\bar{a} = \overline{P(s_{i_1})} \cdots \overline{P(s_{i_k})}\, \overline{V(s_{j_1})} \cdots \overline{V(s_{j_l})}.$$

Note here that our definition prespecifies and fixes the order of the P's and V's for each process a.

We say that $S_S = (\Sigma, N^n, x)$ is the *computation system realized by* S.

THEOREM 6.1. *The class of generalized marked graphs can be simulated by the class of unshared producer–consumer systems with respect to isomorphism.*

Proof. Let $P = (\Pi, \Sigma, \Delta, x)$ be a generalized marked graph, where $\Pi = \{s_1, ..., s_n\}$ and $\Sigma = \{a_1, ..., a_m\}$. Now we construct a producer–consumer system S over semaphores $\{s_1, ..., s_n\}$ as follows. For each i, $1 \leqslant i \leqslant m$. let b_i be the process defined by

$$P(s_1)^{\Delta(s_1, a_i)} \cdots P(s_n)^{\Delta(s_n, a_i)}\, V(s_1)^{\Delta(a_i, s_1)} \cdots V(s_n)^{\Delta(a_i, s_n)}.$$

Let $\rho: D_P \to N^n$ be the function defined by

$$\rho(y) = \langle y(s_1), y(s_2), ..., y(s_n) \rangle.$$

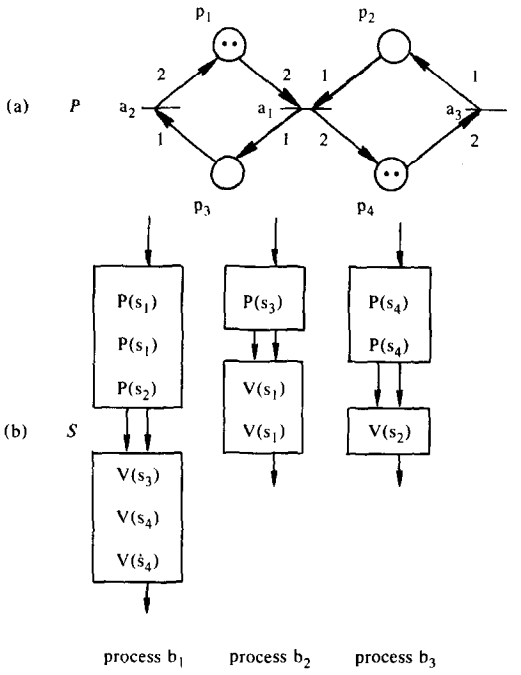


FIG. 6.1. Generalized marked graph P and corresponding unshared producer–consumer system $S = (\{b_1, b_2, b_3\}, \langle 2, 0, 0, 2\rangle)$.

Let $S = (\{b_1,..., b_m\}, \rho(x))$. Note that b_i is a producer of s_j if and only if $\Delta(a_i, s_j) \neq 0$, and b_i is a consumer of s_j if and only if $\Delta(s_j, a_i) \neq 0$. Thus, S is unshared. Let τ: $\{a_1,..., a_m\}^* \to \{b_1,..., b_m\}^*$ be the homomorphism defined by

$$\tau(a_i) = b_i, \qquad 1 \leqslant i \leqslant m.$$

Then, for y and z in D_P and a_i,

$$y \xrightarrow{a_i} z \quad \text{in } P, \quad \text{iff} \quad \rho(y) \xrightarrow{b_i} \rho(z) \quad \text{in } S.$$

Thus, (τ, ρ) is an isomorphism. ∎

EXAMPLE 6.1. Consider the generalized marked graph P depicted in Fig. 6.1(a). The unshared producer–consumer system S isomorphic to P is given in Fig. 6.1b. The region between the P's and V's is the figure could contain other useful computations for the processes.

DEFINITION 6.3. A *computation graph* $G = (V, E, x, U, W, T)$ is a directed graph consisting of:

(i) a finite set V of nodes,

(ii) a finite set E of edges, where any given edge e is directed from a specified node initial(e) to a specified node terminal(e),

(iii) four nonnegative integer functions $x(e)$, $U(e)$, $W(e)$, $T(e)$, where $T(e) \geqslant W(e)$, are associated with each edge e. The function x: $E \to N$ is called the *initial marking*; T: $E \to N$ is called the *threshold function*; U and W are called the *firing functions*.

Informally, a computation of a computation graph is defined as follows: In a computation graph, each edge acts as a queue and each node acts as an operation. For each e, $x(e)$ is the number of items initially in a queue, $T(e)$ is the number of items needed in a queue before its terminal node may be activated, $U(e)$ is the number of items added to the queue whenever its initial node completes an activation, and $W(e)$ is the number of items removed from the queue when its terminal node activates.

A function y: $E \to N$ is called a *marking*. Let D_G be the set of all markings of G. For each $a \in V$, the partial function $\bar{a}$: $D_G \to D_G$ is defined as follows: let $y \in D_G$. Then $\bar{a}(y)$ is defined if and only if $y(e) \geqslant T(e)$ for all edges e directed into a. In this case $\bar{a}(y) = z$ is defined by

$$\begin{aligned} z(e) &= y(e) - W(e), && \text{if} \quad \text{terminal}(e) = a, \quad \text{initial}(e) \neq a, \\ &= y(e) + U(e), && \text{if} \quad \text{terminal}(e) \neq a, \quad \text{initial}(e) = a, \\ &= y(e) - W(e) + U(e), && \text{if} \quad \text{terminal}(e) = a, \quad \text{initial}(e) = a, \\ &= y(e), && \text{otherwise.} \end{aligned}$$

DEFINITION 6.4. A computation graph G is said to be *threshold preserving* if $T(e) = W(e)$ for all edges e of G.

THEOREM 6.2. *The class of unshared producer–consumer systems can be simulated by the class of threshold preserving computation graphs with respect to isomorphisms.*

Proof. Let $S = (\{b_1, \ldots, b_n\}, x)$ be an unshared producer–consumer system. Let $\{s_1, \ldots, s_m\}$ be the set of semaphores appearing in S. Now we construct a computation graph G as follows: The set of nodes of G is $\{a_1, \ldots, a_n\}$, and the set of edges of G is $\{e_1, \ldots, e_m\}$. The initial node and terminal node of e_j are defined by

$$\begin{aligned} \text{initial}(e_j) &= a_i, && \text{if} \quad b_i \text{ is the producer of } s_j, \\ &= a_1, && \text{if} \quad \text{there is no producer of } s_j; \\ \text{terminal}(e_j) &= a_i, && \text{if} \quad b_i \text{ is the consumer of } s_j, \\ &= a_1, && \text{if} \quad \text{there is no consumer of } s_j. \end{aligned}$$

The functions T, U, and W are defined as follows: Suppose that s_j has the producer

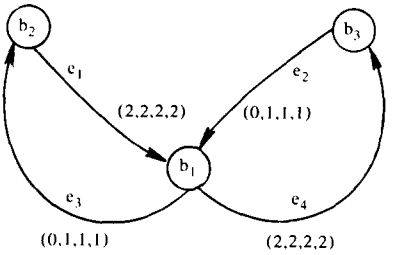


FIG. 6.2. A computation graph isomorphic to S.

b_i. Then $U(e_j)$ is the number of $V(s_j)$ operations in b_i. If s_j has no producer, then $U(e_j) = 0$. Suppose that s_j has the consumer b_i. Then $T(e_j) = W(e_j)$ is the number of $P(s_j)$ operations in b_i. If s_j has no consumer, then $T(e_j) = W(e_j) = 0$. The initial marking x' of G is defined by $x'(e) = x(s_j)$, $1 \leqslant j \leqslant m$. It should be clear that G is isomorphic to S. ∎

EXAMPLE 6.2. The computation graph described in Fig. 6.2 is isomorphic to the unshared producer–consumer system S in Fig. 6.1b, where a label (n_1, n_2, n_3, n_4) attached to edge e stands for $(x(e), V(e), W(e), T(e))$.

LEMMA 6.1. *For each threshold preserving computation graph G, there exists an isomorphic threshold preserving computation graph G' such that*

$$U'(e) \neq 0 \qquad \textit{and} \qquad W'(e) \neq 0$$

for each edge e of G', where U' and W' are the firing functions of G'.

Proof. For each computation graph G, let $\sigma(G)$ be the integer defined by

$$\sigma(G) = \#\{e \mid U(e) = 0\} + \#\{e \mid W(e) = 0\}.$$

We prove the lemma by induction on $\sigma(G)$.

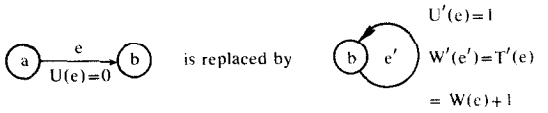
Suppose that there is an edge e of G such that $U(e) = 0$. Let e be directed from node a to node b. Remove e from G and add an edge e' directed from b to b. Let G' be the resulting computation graph where U', W', T', and the initial marking of G' is defined by

$$U'(e') = 1, \qquad W'(e') = T'(e') = W(e) + 1, \qquad x'(e') = x(e) + 1,$$

and for each edge s, $s \neq e$.

$$U'(s) = U(s), \qquad W'(s) = T'(s) = W(s), \qquad x'(s) = x(s).$$

That is,



Clearly, G' is isomorphic to G.

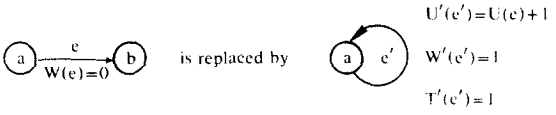
Suppose that there is an edge e of G such that $W(e) = 0$. Let e be directed from a to b. Remove e from G and add an edge e' from a to a. Let G' be the resulting computation graph, where

$$U'(e') = U(e) + 1, \qquad W'(e') = T'(e') = 1, \qquad x'(e') = x(e) + 1,$$

and for s, $s \neq e$,

$$U'(s) = U(s), \qquad W'(s) = T'(s) = W(s), \qquad x'(s) = x(s).$$

That is



Then, it should be clear that G' is isomorphic to G. ∎

THEOREM 6.3. *The class of threshold preserving computation graphs can be simulated by the class of generalized marked graphs with respect to isomorphisms.*

Proof. Let $G = (V, E, x, U, W, T)$ be a threshold preserving computation graph. By Lemma 6.1, we may assume that

$$U(e) \neq 0, \qquad W(e) \neq 0 \qquad \text{for all} \quad e \in E.$$

Let $P = (E, V, \Delta, x)$ be the generalized Petri net constructed as follows: The set of places of P is E, the set of transitions of P is V, and Δ is defined by

$$\begin{aligned} \Delta(v, e) &= U(e), && \text{if} \quad v = \text{initial}(e), \\ &= 0, && \text{otherwise;} \\ \Delta(e, v) &= W(e), && \text{if} \quad v = \text{terminal}(e), \\ &= 0, && \text{otherwise.} \end{aligned}$$

Since $\text{indeg}(e) \leqslant 1$ and $\text{outdeg}(e) \leqslant 1$ for each e in E, P is a generalized marked graph. Clearly, P is isomorphic to G. ∎

The results obtained in this section are summarized in Fig. 6.3, where $C_1 \to C_2$ means that the class C_2 is simulated by the class C_1 with respect to isomorphism.

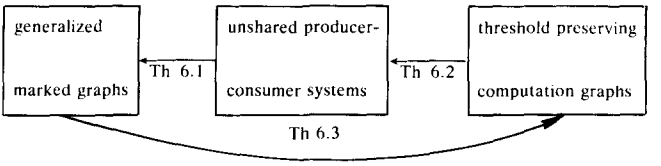


FIG. 6.3. Relationship between computation graphs, generalized marked graphs, and unshared producer–consumer systems.

DEFINITION 6.5. Let $S = (\{a_1,..., a_n\}, x)$ be a producer–consumer system over semaphores $\{s_1,..., s_m\}$. For each i $(1 \leqslant i \leqslant n)$ and j $(1 \leqslant j \leqslant |a_i|)$, let f_{ij} be an abstract symbol. Let $S'_S = (\Sigma', D', x')$ be the computation system defined as follows:

$$\Sigma' = \{f_{ij} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant |a_i|\}, \qquad D' = N^n \times N^m, \qquad x' = \langle 0,..., 0\rangle; x.$$

For each i and j, the partial function $\bar{f}_{ij}$: $D' \to D'$ is defined as follows: Let $y = \langle k_1,..., k_n, v_1,..., v_m\rangle$. Suppose that the jth operation of a_i is $P(s_l)$. Then $\bar{f}_{ij}(y)$ is defined if and only if $k_i = j - 1$ and $v_l > 0$. In this case $\bar{f}_{ij}(y) = \langle k'_1,..., k'_n, v'_1,..., v'_m\rangle$ is defined as follows:

$$\begin{aligned} k'_t &= k_t, && \text{if } t \neq i. \\ &= k_i + 1 \bmod |a_i|, && \text{if } t = i; \\ v'_t &= v_t, && \text{if } t \neq l, \\ &= v_l - 1, && \text{if } t = l. \end{aligned}$$

Suppose that the jth operation of a_i is $V(s_l)$. Then $\bar{f}_{ij}(y)$ is defined if and only if $k_i = j - 1$. In this case $\bar{f}_{ij}(y) = \langle k'_1,..., k'_n, v'_1,..., v'_m\rangle$ is defined by

$$\begin{aligned} k'_t &= k_t, && \text{if } t \neq i, \\ &= k_i + 1 \bmod |a_i|, && \text{if } t = i; \\ v'_t &= v_t, && \text{if } t \neq l, \\ &= v_l + 1, && \text{if } t = l. \end{aligned}$$

The computation system S' is called the *detailed form* of S. ∎

THEOREM 6.4. *Let S be a producer–consumer system. Let S be the computation system realized by S. Let S' be the detailed form of S. If S is unshared, then there exists a spanning homomorphism h: $S \to S'$.*

Proof. Let $S = (\{a_1,..., a_n\}, x)$ be a producer–consumer system over semaphores $\{s_1,..., s_m\}$. For each i $(1 \leqslant i \leqslant n)$, let a_i be of the form

$$a_i = P(s_{i1}) \cdots P(s_{ik_i})\, V(s_{ik_i+1}) \cdots V(s_{il_i}).$$

Then,

$$S = (\Sigma, D, x), \qquad \Sigma = \{a_1,..., a_n\}, \qquad D = N^m,$$
$$S' = (\Sigma', D', x'), \qquad \Sigma' = \{f_{11},..., f_{1l_1},..., f_{n1},..., f_{nl_n}\},$$
$$D' = N^{n+m}, \qquad x' = 0^n; x.$$

Let $h = (\rho, \tau)$ be defined by

$$\rho(y) = 0^n; y \qquad \text{for} \quad y \in N^m,$$
$$\tau(a_i) = f_{i1} f_{i2} \cdots f_{il_i} \qquad \text{for} \quad i = 1, 2,..., n.$$

Now we show that h is a spanning homomorphism. Clearly, h satisfies (2.1) and (2.2), and hence h is a homomorphism. First we note that

$$\text{if } 1 \leqslant i \neq j \leqslant n,\ 1 \leqslant t \leqslant k_i,\ k_j < v \leqslant l_j \text{(i.e., if } f_{it} \text{ is a } P\text{-operation}$$
$$\text{and } f_{jv} \text{ is a } V\text{-operation), then } y \xrightarrow{f_{it} f_{jv}} y' \text{ implies } y \xrightarrow{f_{jv} f_{it}} y'. \tag{6.1}$$

Since S is unshared, if $1 \leqslant i \neq j \leqslant n$, $1 \leqslant t \leqslant k_i$, $1 \leqslant v \leqslant k_j$, then $s_{it} \neq s_{jv}$. Hence

$$\text{if } 1 \leqslant i \neq j \leqslant n,\ 1 \leqslant t \leqslant k_i,\ 1 \leqslant v \leqslant k_j, \text{ then}$$
$$y \xrightarrow{f_{it} f_{jv}} y' \text{ implies } y \xrightarrow{f_{jv} f_{it}} y' \tag{6.2}$$

Furthermore, for $1 \leqslant i, j \leqslant n$, $1 \leqslant t \leqslant l_i$, $1 \leqslant v \leqslant l_j$,

$$\text{if } y \xrightarrow{f_{it}} y',\ y \xrightarrow{f_{jv}} z, \text{ then there exists a } z \in D' \text{ such that}$$
$$y' \xrightarrow{f_{jv}} z' \text{ and } z \xrightarrow{f_{it}} z' \tag{6.3}$$

Let $k = k_1 + k_2 + \cdots + k_n$. Now we show that h satisfies (2.6). Suppose that $\rho(y) \to^{\beta} y'$. Then, by (6.4), there exist β' and $z' = \langle t'_1,..., t'_n, v'_1,..., v'_m \rangle$ such that $0 \leqslant t'_i < k_i$, $i = 1, 2,..., n$ and

$$y' \xrightarrow{\beta'} z'.$$

By (6.5), there exist $\alpha \in \Sigma^*$, $\beta'' \in (\Sigma')^*$, $z \in D$ such that

$$\rho(y) \xrightarrow{\tau(\alpha)} \rho(z) \xrightarrow{\beta''} z', \qquad \tau(\alpha)\beta'' \simeq \beta\beta'$$

and β'' contains no V-operation. Since β'' contains no V-operation,

$$|\beta''| \leqslant k_1 + k_2 + \cdots + k_n = k.$$

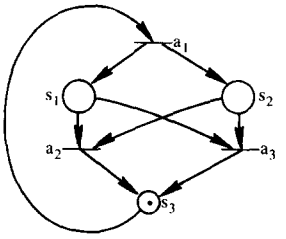


FIG. 6.4. The Petri net isomorphic to S.

By (6.3), it follows that, if $\rho(z) \to^{\gamma} \rho(u)$, then there exist γ', γ'', and u' such that

$$z' \xrightarrow{\gamma'} u', \qquad \rho(u) \xrightarrow{\gamma''} u', \qquad \tau(\gamma)\gamma'' \simeq \beta''\gamma'.$$

Hence (2.6) holds.

Suppose that $\rho(y) \to^{\beta} \rho(w) = \langle 0,...,0, v_1,..., v_m \rangle$. Then, by (6.5), there exist $z \in D$, $\beta'' \in (\Sigma')^*$, and $\alpha \in \Sigma^*$ such that $\tau(\alpha)\beta'' \simeq \beta$,

$$\rho(y) \xrightarrow{\tau(\alpha)} \rho(z) \xrightarrow{\beta''} \rho(w),$$

and β'' contains no V-operation. Then $\beta'' = \lambda$ and $\rho(z) = \rho(w)$. Therefore (2.5) holds. Hence h is a spanning homomorphism. ∎

Note that the homomorphism constructed in the proof of Theorem 6.4 satisfies the condition of Lemma 4.1. Hence liveness is preserved by this homomorphism.

It should be noted that Theorem 6.4 does not hold when S is not unshared. For example, consider the three process producer–consumer system:

$$S = (\{P(s_3)\,V(s_1)\,V(s_2), P(s_1)\,P(s_2)\,V(s_3), P(s_2)\,P(s_1)\,V(s_3)\}, \langle 0, 0, 1 \rangle).$$

The computation system S realized by S is isomorphic to the Petri net of Fig. 6.4. The detailed form S' of S is shown in Figure 6.5.

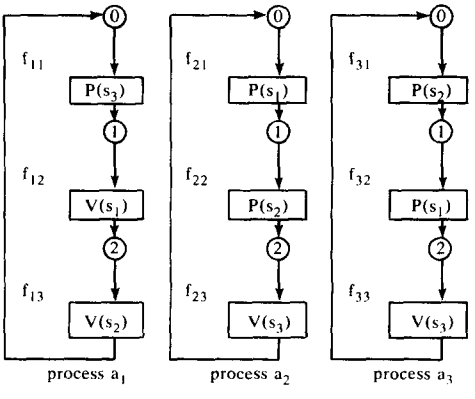


FIG. 6.5. The detailed form S' of S.

Here, S' has a deadlock. In fact, $f_{11}f_{12}f_{13}f_{21}f_{31}f \notin C_{S'}$ for all $f \in \{f_{ij} \mid 1 \leqslant i, j \leqslant 3\}$. No deadlock, however, occurs in the Petri net S. Hence there is no spanning homomorphism from S to S'. This example is taken from [16], in which this deadlock anomaly was noted. Here, however, we see that the spanning homomorphism provides the appropriate formulation for relating Petri nets and producer–consumer systems.

Next, we note that

$$\text{if } \rho(y) \xrightarrow{\beta} y' = \langle t_1,..., t_n, v_1,..., v_m\rangle, \text{ then there exist } \beta', t'_1,..., t'_n, v'_1,..., v'_m \text{ such that}$$

$$y' \xrightarrow{\beta'} \langle t'_1,..., t'_n, v'_1,..., v'_m\rangle, \qquad 0 \leqslant t'_i < k_i, \quad i = 1, 2,..., n. \tag{6.4}$$

In fact, if $t_i \geqslant k_i$, then all of $f_{it_i+1},..., f_{il_i}$ are V-operations, and hence

$$y' \xrightarrow{f_{it_i+1},\cdots,f_{il_i}} \langle t_1,..., t_{i-1}, 0, t_{i+1},..., t_n, v'_1,..., v'_m\rangle$$

for some $v'_1,..., v'_m$.

Suppose that $\rho(y) \to^{\beta\beta'} z' = \langle t'_1,..., t'_n, v'_1,..., v'_m\rangle$ and $0 \leqslant t'_i < k_i$, $i = 1, 2,..., n$. Next we show that

$$\text{if } \beta\beta' \text{ contains a } V\text{-operation, then there exist } a \in \Sigma, z \in D, \beta'' \in (\Sigma')^* \text{ such that } p(y) \xrightarrow{\tau(a)} \rho(z) \xrightarrow{\beta''} z', \tau(a)\beta'' \simeq \beta\beta'. \tag{6.5}$$

Suppose that $\beta\beta'$ contains a V-operation. Then $\beta\beta'$ must be of the form

$$\beta\beta' = \beta_1 f_{i1}\beta_2 f_{i2} \cdots f_{ik_i+1} \cdots f_{il_i}\beta_{l_i+1},$$

where $\beta_1,..., \beta_{l_i+1} \in (\Sigma')^*$ and f_{ik_i+1} is the leftmost occurrence of a V-operation in $\beta\beta'$. Then, by (6.1) and (6.2), we have

$$\rho(y) \xrightarrow{f_{i1}f_{i2}\cdots f_{il_i}} \rho(z) \xrightarrow{\beta_1\beta_2\cdots\beta_{l_i+1}} z'$$

for some $z \in D$. Hence (6.5) holds.

7. Petri Nets and Generalized Petri Nets

In this section, we consider relationships between Petri nets and generalized Petri nets using our notions of homomorphisms, simulation, and emulation. Since Petri nets are a subclass of generalized Petri nets, it is obvious that generalized Petri nets can simulate Petri nets, and that any technique used to solve questions about generalized Petri nets can be directly applied to questions about Petri nets also.

The simulation of emulation of generalized Petri nets by Petri nets is not so obvious. What we do in this section is show that, for any transition a and place p, the $\Delta(a, p)$ function of a generalized Petri net can be iteratively reduced to value 1 as required for Petri nets. Then we show that $\Delta(p, a)$ can also be reduced, and this provides us with the result that a generalized Petri net can be emulated by a Petri net with respect to a principal homomorphism. Although relationships between Petri nets and generalized Petri nets have been investigated by others previously, especially Hack, it seems that our results, at least our constructions, differ significantly from those done previously. Further reference to this work can be found in [19].

DEFINITION 7.1. Let $P = (\Pi, \Sigma, \Delta, x)$ be a generalized Petri net. For each $a \in \Sigma$, let

$$\text{out-index}(a) = \sum_{\Delta(a,p)>1} (\Delta(a, p) - 1), \qquad \text{in-index}(a) = \sum_{\Delta(p,a)>1} (\Delta(p, a) - 1).$$

The *out-index* and *in-index* of P is defined by

$$\text{out-index}(P) = \sum_{a \in \Sigma} \text{out-index}(a), \qquad \text{in-index}(P) = \sum_{a \in \Sigma} \text{in-index}(a).$$

Note that P is a Petri net if and only if $\text{out-index}(P) = \text{in-index}(P) = 0$.

LEMMA 7.1. *A generalized Petri net $P = (\Pi, \Sigma, \Delta, x)$ of out-index n, $n > 0$, can be simulated by a generalized Petri net P' of out-index $n - 1$ with respect to a principal homomorphism.*

Proof. Let $a \in \Sigma$ be a transition such that $\text{out-index}(a) \geqslant 1$. Then there exists $p \in \Pi$ such that $\Delta(a, p) > 1$.

Given transition a of P, then P' is essentially constructed as follows: Transition a is replaced by transitions a_1 and a_2 which are controlled so as to fire in sequence $a_1 a_2$ in P' as a replacement for the firing of a in P. The $\Delta(a, p)$ of P can thus be reduced by 1 by letting $\Delta'(a_1, p) = 1$ and $\Delta'(a_2, p) = \Delta(a, p) - 1$, thus causing the number of tokens deposited in p by the firing of a to be identical to the number deposited by the $a_1 a_2$ firing in P'. For control purposes, two new places q_1 and q_2 are used in P', q_2 to sequence the $a_1 a_2$ firing and q_1 to insure that new firing sequences cannot arise in P' due to the splitting of transition a. The details of the construction are as follows: Let $P' = (\Pi', \Sigma', \Delta', x')$ be the generalized Petri net defined by

$$\Pi' = \Pi \cup \{q_1, q_2\}, \qquad \Sigma' = (\Sigma - \{a\}) \cup \{a_1, a_2\}$$

$$x'(s) = x(s), \quad \text{if } s \in \Pi,$$
$$= 1, \quad \text{if } s = q_1,$$
$$= 0, \quad \text{if } s = q_2.$$

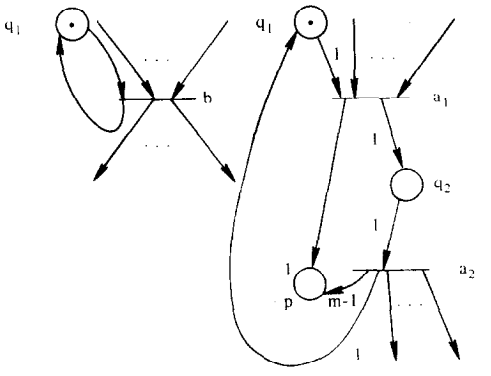


FIG. 7.1. $b \in \Sigma - \{a\}$, $m = \Delta(a, p)$.

We define Δ' as follows: (see Fig. 7.1)

(i) For $s \in \Pi'$ and $b \in \Sigma - \{a\}$

$$\Delta'(s, b) = \Delta(s, b), \qquad \Delta'(b, s) = \Delta(b, s),$$
$$\Delta'(q_1, b) = 1, \qquad \Delta'(q_2, b) = 0,$$
$$\Delta'(b, q_1) = 1, \qquad \Delta'(b, q_2) = 0,$$

(ii) For $s \in \Pi'$ and $a_1 \in \Sigma'$

$$\Delta'(s, a_1) = \Delta(s, a) \qquad \text{for} \quad s \in \Pi,$$
$$\Delta'(q_1, a_1) = 1, \qquad \Delta'(q_2, a_1) = 0;$$
$$\Delta'(a_1, s) = 1, \qquad \text{if} \quad s = p \quad \text{or} \quad s = q_2,$$
$$= 0, \qquad \text{otherwise.}$$

(iii) For each $s \in \Pi'$ and $a_2 \in \Sigma'$

$$\Delta'(s, a_2) = 1, \qquad \text{if} \quad s = q_2,$$
$$= 0, \qquad \text{otherwise.}$$
$$\Delta'(a_2, s) = 1, \qquad \text{if} \quad s = q_1,$$
$$= 0, \qquad \text{if} \quad s = q_2,$$
$$= \Delta(a, p) - 1, \qquad \text{if} \quad s = p,$$
$$= \Delta(a, s), \qquad \text{otherwise.}$$

Let τ: $\Sigma^* \to (\Sigma')^*$ be the homomorphism defined by

$$\tau(b) = b, \qquad b \in \Sigma - \{a\}, \qquad \tau(a) = a_1 a_2$$

Let ρ: $DP \to DP'$ be the function defined by

$$\begin{aligned} \rho(y)(s) &= y(s), && \text{if } s \in \Pi, \\ &= 1, && \text{if } s = q_1, \\ &= 0, && \text{if } s = q_2. \end{aligned}$$

Clearly, $h = (\tau, \rho)$ satisfies conditions (2.1) and (2.2), and hence, h is a homomorphism. Suppose that for $y \in DP$ and $z \in DP$,

$$\rho(y) \xrightarrow{a_1} a$$

holds in P'. Then $z(q_1) = 0$ and $z(q_2) = 1$. Since $z(q_2) = 1$, the transition a_2 is fireable at z. Since $z(q_1) = 0$, no transition other than a_2 is fireable at z. Hence (2.3) and (2.4) hold. Since $\iota(\tau(b)) \cap \iota(\tau(c)) = \varnothing$ for $b \neq c$, h is a principal homomorphism. ∎

By repeated application of Lemma 7.1 we obtain the following corollary.

COROLLARY 7.1. *The class of generalized Petri nets can be simulated by the class of generalized Petri nets of out-index* 0 *with respect to principal homomorphisms.*

The next logical step would be to find a construction to reduce the in-index of a generalize Petri net with respect to a principal or spanning homomorphism so that one could simulate the generalized Petri net by a Petri net with respect to such a homomorphism. We have not been successful in finding such a construction. On the other hand, we have not shown that this is impossible. Thus we leave this as an open problem worth considering.

Our next lemma, however, shows how to split the edge weight $\Delta(p, a)$ approximately in half. Thus, it enables us to study the properties of generalized Petri nets by Petri nets through emulation using surjective homomorphisms.

LEMMA 7.2. *The class of generalized Petri nets of out-index* 0 *can be emulated by the class of Petri nets with respect to surjective homomorphisms.*

Proof. Let $P = (\Pi, \Sigma, \Delta, x)$ be a generalized Petri net of out-index 0 and in-index m. To prove the lemma, it suffices to construct a generalized Petri net P' of out-index 0 and in-index less than m, and a surjective homomorphism from P' to P. We construct P' as follows: Let $P' = (\Pi', \Sigma', \Delta', x')$, where

$$\Pi' = \Pi \cup \{q_0, q_1\}, \qquad \Sigma' = \{a_0, a_1 \mid a \in \Sigma\}.$$

Since P has in-index $m > 0$, it must contain some place p with $\Delta(p, a) > 1$. We pick such a place p and construct P' based on this choice of p. Let

$$
\begin{aligned}
x'(s) = x(s), \quad & \text{if} \quad s \in \Pi - \{p\}, \\
= \left\lfloor \frac{x(p)}{2} \right\rfloor, \quad & \text{if} \quad s = p, \\
= 1, \quad & \text{if} \quad s = q_0 \quad \text{and} \quad x(p) \text{ is even or} \\
& \qquad s = q_1 \quad \text{and} \quad x(p) \text{ is odd}, \\
= 0, \quad & \text{otherwise.}
\end{aligned}
$$

For each $s \in \Pi - \{p\}$ and a_i ($a \in \Sigma$, $i = 0, 1$), Δ' is defined by

$$\Delta'(s, a_i) = \Delta(s, a), \qquad \Delta'(a_i, s) = \Delta(a, s).$$

For each $a \in \Sigma$ and $s \in \{p, q_0, q_1\}$. $\Delta'(a_i, s)$ and $\Delta'(s, a_i)$, $i = 0, 1$, are defined as in Figs. 7.2–7.5 according to the following four cases: Case (1), $\Delta(p, a) = 2l$, $l \geqslant 0$, $\Delta(a, p) = 0$; Case (2), $\Delta(p, a) = 2l$, $l \geqslant 0$, $\Delta(a, p) = 1$; Case (3), $\Delta(p, a) = 2l + 1$, $l \geqslant 0$, $\Delta(a, p) = 0$; Case (4), $\Delta(p, a) = 2l + 1$, $l \geqslant 0$, $\Delta(a, p) = 1$.

Since this construction splits the $\Delta(p, a)$ of P into two parts $\Delta'(p, a_1)$ and $\Delta'(p, a_2)$ of P' for each transition a of P, using in all cases the same control places q_0 and q_1, this creates a complex of arrows entering q_0 and q_1. We shall see later that q_0 will contain a token if and only if p contains an even number of tokens and q_1 will contain a token if and only if p contains an odd number of tokens. Thus q_0 and q_1 control whether a_0 or a_1 can fire, depending upon the evenness or oddness of tokens in p.

Let τ: $(\Sigma')^* \to \Sigma^*$ be the homomorphism defined by $\tau(a_1) = \tau(a_2) = a$ for each $a \in \Sigma$. Let ρ: $DP' \to DP$ be the function defined by

$$
\begin{aligned}
\rho(y)(s) = y(s), \quad & \text{if} \quad s \in \Pi - \{p\}, \\
= 2y(s) + y(q_1), \quad & \text{if} \quad s = p.
\end{aligned}
$$

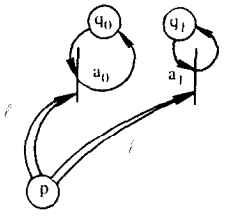


FIG. 7.2. Case 1: $\Delta(p, a) = 2l$, $\Delta(a, p) = 0$.

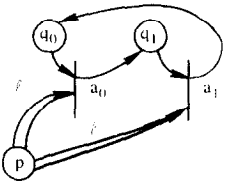


FIG. 7.3. Case 2: $\Delta(p, a) = 2l$, $\Delta(a, p) = 1$.

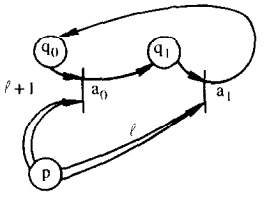


FIG. 7.4. Case 3: $\Delta(p, a) = 2l + 1$, $\Delta(a, p) = 0$.

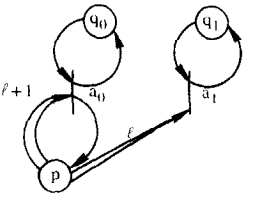


FIG. 7.5. Case 4: $\Delta(p, a) = 2l + 1$, $\Delta(a, p) = 1$.

Now we show that $h = (\tau, \rho)$ is a surjective homomorphism. First note that if

$$x' \xrightarrow{*} y \qquad \text{in} \quad P',$$

then $y(q_0) + y(q_1) = 1$, that is, there is exactly one token on either q_0 or q_1. Furthermore, transition a_0 in Σ' is fireable at y if and only if $y(q_0) = 1$, and a_1 is firable at y if and only if $y(q_1) = 1$.

To prove that h is a surjective homomorphism, it suffices to show that

$$\begin{aligned} &x' \xrightarrow{a} y \qquad \text{in} \quad P' \qquad \text{if and only if} \\ &\rho(x') = x \xrightarrow{\tau(a)} \rho(y) \qquad \text{in} \quad P. \end{aligned} \tag{7.1}$$

We prove (7.1) by induction on the length of α. It $\alpha = \lambda$, then (7.1) clearly holds. Suppose that (7.1) holds for α. Note that $y(q_1) = 1$ if and only if $\rho(y)(p)$ is odd. Suppose that $\Delta(p, a) = 2l + 1$, $\Delta(a, p) = 0$ (Case (3)) and $i = 0$. (We only prove the result for this case, the other cases can be proved similarly.) Then a_0 is fireable at y if and only if

$$\begin{aligned} &y(s) \geqslant \Delta'(s, a_0) = \Delta(s, a), \qquad s \in \Pi - \{p\}, \\ &y(q_0) = \Delta'(q_0, a_0) = 1, \qquad y(p) \geqslant \Delta'(p, a_0) = l + 1 \end{aligned}$$

if and only if

$$\rho(y)(s) = y(s) \geqslant \Delta(s, a), \qquad s \in \Pi - \{p\}; \qquad \rho(y)(p) = 2y(p) \geqslant \Delta(p, a) = 2l + 1$$

if and only if a is fireable at $\rho(y)$. Suppose that a_0 is fireable at y, and let

$$y \xrightarrow{a_0} z, \qquad \rho(y) \xrightarrow{a} z'.$$

Then, for $s \in \Pi - \{p\}$,

$$z'(s) = \rho(y)(s) - \Delta(s, a) + \Delta(a, p) = \rho(z)(s).$$

Since $z(q_1) = 1$,

$$\begin{aligned} z'(p) &= \rho(y)(p) - \Delta(p, a) = 2y(p) - (2l + 1) = 2(y(p) - (l + 1)) + 1 \\ &= 2z(p) + z(q_1) = \rho(z)(p). \end{aligned}$$

Therefore $z' = \rho(z)$, so that (7.1) holds. ∎

Note that the homomorphism constructed in the proof of Lemma 7.2 is not principal, since a_0 and a_1 realize different partial functions, but $\tau(a_0) = \tau(a_1)$. In Petri nets, it is not allowed that two distinct transitions have the same name. Let us extend the notion of Petri net to allow that distinct transitions can possess the same name (in this case, each $a \in \Sigma$, in general, realizes a binary relation on D but not a partial

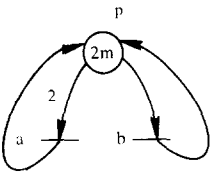


FIG. 7.6. A generalized Petric net P of out-index 0.

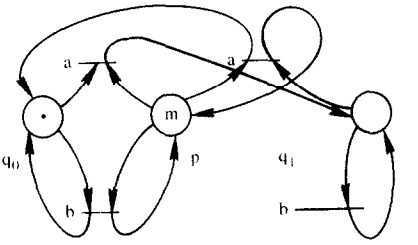


FIG. 7.7. A Petri net P'' which emulates P.

function from D to D). Let P'' be the Petri net obtained from P' by replacing the names a_0 and a_1 by a for each $a \in \Sigma$. Let τ' be the identity homomorphism. Then $h' = (\rho, \tau')$ is a principal homomorphism from P'' to P.

EXAMPLE 7.1. Consider the generalized Petri net P in Fig. 7.6. Then the Petri net P'' in Fig. 7.7 emulates P with respect to a principal homomorphism. Note that, in P'', two upper transitions and two lower transitions have the same names.

Note that any problem in generalized Petri nets mentioned in Section 4 can be reduced to the corresponding problem in Petri nets. In fact, by Corollary 7.1, we can reduce it to the problem in generalized Petri nets of out-index 0. Then, by Lemma 7.2, we can reduce it to the problem in Petri nets.

8. CONCLUSIONS

In our quest for providing a firm mathematical basis for relating different models of parallel computation, we have introduced an underlying model that we call a computation system. We showed how various other models (namely, vector addition systems, vector replacement systems, Petri nets, generalized Petri nets, computation graphs, unshared producer-consumer systems, and generalized marked graphs) could be realized in terms of computation systems; and then we related the various models through this common formulation. In order to provide precise relationships, we introduced the concept of homomorphisms between computation systems. In so doing we found that different types of homomorphisms were advisable to introduce. This was so since the various properties that one was interested in capturing and studying within the models turn out to be preserved only under certain homomorphisms. These details are given in Section 4.

As we developed the computation system model, we became aware of other related work, e.g., that of Kwong [12] and Gourlay *et al.* [5], that had some similarities with our work; particularly in the sense that they also introduced mappings that preserved properties. We have attempted to provide some insight into how these other works are related to ours by making some brief comments on this in the first few sections of the paper.

This study, although extensive, is incomplete in several ways. First, there are other properties and models of parallel computation, and variations of the models, that we have not touched on here. We believe, however, that the computation system approach we have introduced here could be used to study other properties and models as well. Second, there are other papers that provide discussions of relationships between models of parallel computation—often treated in a much different manner, and using different constructions than we have used. One such example is that of Lipton *et al.* [13]. We have not attempted to mold these other approaches into our computation system and homomorphism approach. In fact, it is not clear how, or if, this could be done. It is certainly possible that the types of questions asked about the models, and the properties studied, are sufficiently different so as to make any such comparison of little use. Finally, there are some open questions that could be looked at further. One is the capturing of the various notions of lockout, starvation, or unbounded waiting in terms of computation systems, and the determination of what types of homomorphisms preserve these properties. Another is whether a direct construction can be found which reduces the in-index of generalized Petri nets with respect to a homomorphism that preserves the desired properties. Such a result could then provide a way to directly simulate generalized Petri nets by Petri nets. One approach here, would be to look at constructions given by other authors (e.g., Hack) in their works at relating Petri nets and generalized Petri nets to see if, or how, these approaches fit into the computation system and homomorphism approach that we have introduced.

REFERENCES

1. J. L. BAER, A survey of some theoretical aspects of multiprocessing, *Comput. Surveys* **5** (1) (1973), 31–80.
2. T. H. BREDT, "A Survey of Models for Parallel Computing," Stanford Univ. Elect. Lab. Tech. Report 8, Stanford, Ca., August 1970.
3. T. H. BREDT AND E. J. MCCLUSKY, Analysis and synthesis of control mechanisms for parallel processes *in* "Parallel Processor Systems Technologies and Applications" (L. C. Hobbs *et al.*, Eds.), pp. 287–296, Washington, D.C., 1970.
4. E. W. DIJKSTRA, Cooperating sequential processes, *in* "Programming Languages" (F. Genuys, Ed.), pp. 43–112, Academic Press, New York, 1968.
5. J. S. GOURLAY, W. C. ROUNDS, AND R. STATMAN, On properties preserved by contractions of concurrent systems, *in* "Semantics of Concurrent Computation," Lecture Notes in Computer Science, No. 70, pp. 51–65, Springer-Verlag, Berlin/New York, 1979.
6. M. HACK, The equivalence problem for vector addition systems is undecidable, *Theoret. Comput. Sci.* **2** (1976), 77–95.
7. M. HACK, The recursive equivalence of the reachability problem and the liveness problem for Petri nets and vector addition systems, *in* "Proceedings of the 15th Annual IEEE Symposium on Switching and Automata Theory," pp. 145–155, October 1974.
8. R. M. KARP AND R. E. MILLER, Properties of a model for parallel computations; determinacy, termination, queueing, *SIAM J.* **14** (6) (1966), 1390–1411.
9. R. M. KARP AND R. E. MILLER, Parallel program schemata, *J. Comput. System Sci.* **3** (1969), 147–195.

10. R. M. KELLER, "Vector Replacement Systems: A Formalism for Modelling Asynchronous Systems," Princeton Univ. E.E. Tech. Report No. 117, Princeton, N.J., December 1972, revised January 1974.
11. R. M. KELLER, Formal verification of parallel programs, *Comm. ACM* **19** (1976), 371–374.
12. Y. S. KWONG, On reduction of asynchonous systems, *Theoret. Comput. Sci.* **5** (1977), 25–50.
13. R. J. LIPTON, L. SNYDER, AND Y. ZALCSTEIN, A comparative study of parallel computation, *in* "Proceedings of the 15th Annual IEEE Symposium on Switching and Automata Theory," pp. 145–155, October 1974.
14. R. J. LIPTON, Reduction: A method of proving properties of parallel programs, *Comm. ACM* **12** (1975), 717–721.
15. R. E. MILLER, A comparison of some theoretical models of parallel computation, *IEEE Trans. Comput.* **C-22** (8) (1973), 710–717.
16. R. E. MILLER, "Relationships Among Models of Parallelism and Synchronization," presented at the Symposium on Petri Nets and Related Methods, M.I.T., Cambridge, Mass., July 1–3, 1975.
17. R. E. MILLER, Theoretical studies of asynchronous and parallel processing, *in* "Proceedings of the 1977 Conference on Information Sciences and Systems," pp. 333–339, Johns Hopkins University, March, 1977.
18. J. L. PETERSON AND T. H. BREDT, A comparison of models of parallel computation, *in* "Information Processing 74, Proceedings, IFIP Congress," pp. 466–470, North-Holland, Amsterdam/London 1974.
19. J. L. PETERSON, Petri nets, *Comput. Surveys* **9** (1977), 223–252.
20. C. A. PETRI, "Communication with Automata," Supplement 1 to Tech. Report RADC-TR-65-337, Vol. 1, Griffiss Air Force Base, New York, 1966. (Translated from "Kommunikation mit Automata," Univ. Bonn, Germany 1962.)